UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO DANIEL AGOSTINI

UMA ANÁLISE EXPERIMENTAL DE ABORDAGENS TOPOLÓGICAS APLICADAS AO PROBLEMA DO CAIXEIRO-VIAJANTE ATRAVÉS DE OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS

CURITIBA

2015

BRUNO DANIEL AGOSTINI

UMA ANÁLISE EXPERIMENTAL DE ABORDAGENS TOPOLÓGICAS APLICADAS AO PROBLEMA DO CAIXEIRO-VIAJANTE ATRAVÉS DE OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS

Dissertação apresentada ao Programa de Pós Graduação em Métodos Numéricos em Engenharia, área de concentração em Programação Matemática, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de mestre em Métodos Numéricos.

Orientador: Prof. Dr. Paulo Henrique Siqueira

CURITIBA

2015

Agostini, Bruno Daniel
Uma análise experimental de abordagens topológicas aplicadas ao problema do caixeiro viajante através de otimização por nuvem de partículas / Bruno Daniel Agostini. – Curitiba, 2015.
89 f. : il.; tabs.

Dissertação (mestrado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Matemática.
Orientador: Paulo Henrique Siqueira
Bibliografia: p. 85-89

1. Otimização combinatória. 2. Análise combinatória. 3. Algori - tmos. I. Siqueira, Paulo Henrique. II. Título

CDD 519.64

TERMO DE APROVAÇÃO

BRUNO DANIEL AGOSTINI

UMA ANÁLISE EXPERIMENTAL DE ABORDAGENS TOPOLÓGICAS APLICADAS AO PROBLEMA DO CAIXEIRO VIAJANTE ATRAVÉS DE OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS

Dissertação aprovada como requisito parcial para obtenção do grau de mestre no Programa de Pós-Graduação em Métodos Numéricos em Engenharia da Universidade Federal do Paraná, pela seguinte banca examinadora:

Prof. Dr. Paulo Henrique Siqueira.
(Orientador) Membro do PPGMNE/UFPR

Prof.ª Dr.ª Ângela Olandoski Barboza.
Membro da UTFPR

Prof.ª Dr.ª Deise Maria Bertholdi Costa.
Membro do PPGMNE/UFPR

Curitiba, 25 de abril 2014.

Resumo

O algoritmo do Particle Swarm Optimization (PSO), inspirado em comportamentos sociais naturais é uma metaheurística que tem sido aplicada com sucesso na resolução de problemas de otimização combinatória. Este trabalho tem como objetivo apresentar o algoritmo PSO com busca local e *path relinking,* que tem apresentados resultados promissores, porém se diferencia de versões anteriores, pois incorpora duas diferentes estratégias. A primeira estratégia se refere à forma de comunicação entre as partículas, sendo propostas as topologias Focal, Von Neumann e Clan. A qualidade das topologias propostas são testadas na comparação com as topologias Global e Local, geralmente utilizadas no algoritmo PSO. A segunda estratégia é a dispersão da nuvem e tem como objetivo procurar melhores regiões no espaço de busca. Tal proposta é validade na aplicação ao Problema do Caixeiro-Viajante.

Abstract

The Particle Swarm Optimization algorithm (PSO), inspired by natural social behavior is a metaheuristic that has been successfully applied in solving combinatorial optimization problems. This work aims to present the PSO algorithm using local search and path relinking, which has shown promising results, but differs from earlier versions, it incorporates two different strategies. The first strategy refers to the means of communication between the particles being proposed Focal, Von Neumann and Clan topologies. The quality of the proposed topologies are tested in comparison with the global and local topologies, typically used in the PSO algorithm. The second strategy is the dispersion of the cloud and aims to seek better regions in the search space. This proposal is valid in application to the Traveling Salesman Problem.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

ABREVIATURAS

ACO	*Ant Colony Optimization*
AG	Algoritmos Genéticos
AS	*Ant System*
CO	*Combinatorial Optimization*
COP	*Combinatorial Optimization Problems*
DP	Desvio Padrão
GRASP	*Greedy Randomized Adaptative Search Procedure*
LC	Lista de Candidatos
LK	Lin-Kernighan
LRC	Lista Restrita de Candidatos
LT	Lista Tabu
PCV	Problema do Caixeiro-Viajante
PMX	*Partially Mapped Crossover*
PSO	*Particle Swarm Optimization*
PSO-PR	*Particle Swarm Optimization with Path Relinking*
TSP	*Traveling Salesman Problem*
TSPLIB	*Traveling Salesman Problem Library*

SUMÁRIO

1. INTRODUÇÃO

1.1 DESCRIÇÃO DO PROBLEMA

Problemas de alta complexidade e inúmeras soluções são encontrados em várias situações reais. Porém encontrar uma solução para esses problemas pode ser uma tarefa de alto custo. Os problemas de otimização de forma geral têm como objetivo maximizar ou minimizar uma função numérica de várias variáveis, definida em seu domínio, definido por um conjunto de restrições.

Entre esses problemas tem-se os de otimização combinatória cujo domínio é finito, isto é, os elementos de seu domínio podem ser testados. Mesmo assim, na prática, torna-se inviável encontrar todos os elementos do domínio na busca pelo melhor. Alguns dos problemas de otimização combinatória que podem ser citados são: o problema do caixeiro-viajante (PCV), o qual será abordado nesta dissertação, o problema da mochila, o da árvore geradora mínima, o da cobertura mínima por conjuntos, o da localização de centros de distribuição e o do roteamento de veículos. Devido à alta complexidade e custo computacional em problemas de grandes dimensões, aliadas à falta de robustez das técnicas clássicas de otimização, a resolução destes problemas impulsionam a busca por novas estratégias.

Com o desenvolvimento da tecnologia computacional, processos que antes envolviam grande tempo para desenvolvimento de cálculos matemáticos, exaustivos e complexos e suscetíveis a falhas, agora podem ser implementados computacionalmente, possibilitando avanços nas técnicas de otimização. Métodos heurísticos e metaheurísticos tem mostrado avanços na obtenção de soluções próximas do ótimo ou por vezes encontrando a solução ótima.

Os métodos metaheurísticos, têm o objetivo de encontrar uma solução viável e eventualmente ótima. Algoritmos Evolutivos, Busca Tabu, Simulated Annealing são algumas das metaheurísticas mais conhecidas. Devem-se destacar os algoritmos evolutivos, especialmente baseados em população, pois recentemente mostram êxito em solucionar problemas com múltiplas soluções (KNOWLES; CORNE, 2000; LAUMANNS; ZITZLER; THIELE, 2000; ZITZLER; DEB; THIELE, 2000).

Atualmente um algoritmo que tem recebido grande destaque é o denominado Otimização por Enxame de Partículas ou *Particle Swarm Optimization* (PSO) (KENNEDY; EBERHART, 2001).

O algoritmo PSO é baseado no princípio de que um grupo de indivíduos possibilita uma busca muito mais abrangente do que apenas um único indivíduo. Desenvolvido inicialmente James Kennedy e Russel Eberhart na década 90 (CARVALHO, 2008), o algoritmo baseia-se na modelagem do comportamento social de bandos de pássaros.

Desde a introdução do *PSO*, pesquisadores propuseram modificações no algoritmo original buscando um melhor desempenho. Essas modificações podem ser feitas alterando-se a sua convergência, incluindo restrições de forma a evitar mínimos locais ou alterando a sua topologia, que é a forma de comunicação das partículas.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

A presente dissertação de mestrado tem como objetivo geral o estudo do algoritmo de Otimização por Enxame de Partículas através de diversas abordagens topológicas discretas aplicadas a instâncias do problema de otimização discreta do caixeiro-viajante, disponibilizadas na biblioteca TSPLIB (*Traveling Salesman Problem Library*). Além de apresentar a estratégia de dispersão da nuvem a fim de ficar evitar mínimos locais.

1.2.2 Objetivos Específicos

Estudar os principais conceitos, heurísticas e metaheurísticas existentes para a resolução do problema.

Adaptar as topologias Focal, Von Neumann e Clan para espaços discretos.

Comparar Focal, Von Neumann e Clan com as topologias Global e Local, já adaptadas a espaços discretos.

1.3 ESTRUTURA DO TRABALHO

Esta dissertação esta organizada em capítulos, detalhados a seguir.

O capítulo 2 apresenta o conceito de roteirização de veículos e descreve o Problema do Caixeiro-Viajante (PCV), com suas variações e formulação matemática. Os capítulos 3, 4 e 5 tratam da fundamentação teórica para o desenvolvimento deste trabalho, onde serão apresentados os conceitos de otimização, algumas das técnicas heurísticas, metaheurísticas utilizada na resolução do PCV, além de apresentar a inteligência por enxames e a otimização por nuvem de partículas em sua estrutura clássica, seu algoritmo e as estruturas de comunicação do enxame de partículas, que são foco de trabalho dessa dissertação. Os capítulos 6 se destina a apresentação do PSO Discreto, e do modelo PSO-PR (*Particle Swarm Optimization with Path-Relinking*), o qual será implementado.

O capitulo 7 é destinado a implementação e apresentação e análise dos resultados dos experimentos computacionais. O capítulo 8 apresenta as considerações finais sobre este trabalho.

2. ROTEIRIZAÇÂO DE VEÍCULOS

De acordo com Malaquias (2006), o termo roteirização deriva do inglês *routing*, para designar o processo de terminação de uma ou mais sequencias ou roteiros de paradas a serem cumpridos por veículos, com o objetivo de visitar um conjunto de pontos pré-determinados, que necessitam de atendimento.

Os problemas de roteamento tratam em sua maior parte com a escolha de rotas sobre os pontos de demanda ou oferta. Esses pontos podem representar cidades, centros de distribuição, depósitos, pontos de coleta, etc. Dentre os tipos de rotas, um dos mais importantes é denominado hamiltoniano. Seu nome é devido a William Rowan Hamilton que em 1857 propôs um jogo chamado *Around the World* (Figura 1).

FIGURA 1: JOGO AROUND THE WORLD

Fonte: http://mathhmagic.blogspot.com.br/2011/05/icosian-game-tetris.html

O jogo consistia em um tabuleiro com um dodecaedro desenhado e cada vértice correspondia a uma cidade importante da época, o desafio era escolher uma rota que passasse por todas as cidades uma única vez, iniciando e terminando na mesma cidade. A solução do seu jogo (Figura 2) passou a se chamada de ciclo hamiltoniano, em sua homenagem.

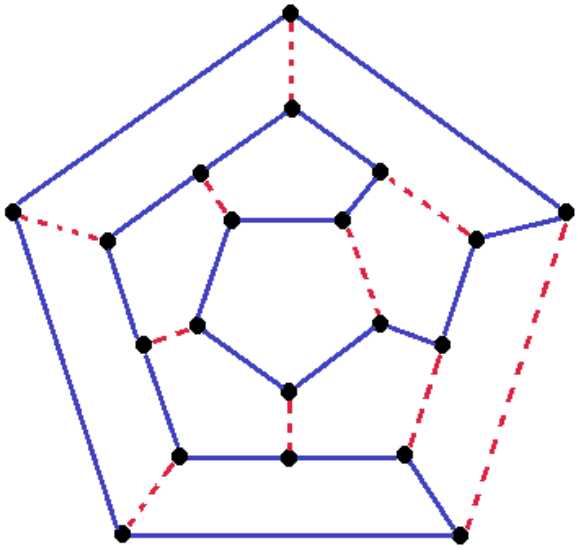

FIGURA 2: SOLUÇÃO DO JOGO *AROUND THE WORLD*

Fonte: O Autor (2014)

Segundo Bodin *et al* (1983), os problemas de roteirização dividem-se em três tipos;

- **Roteirização pura**: não existem restrições temporais, nem de procedência entre os clientes, considerando-se apenas aspectos espaciais.
- **Programação de veículos**: há restrição de horários pré-estabelecidos para cada atividade a ser executada, considerando-se tanto aspectos temporais quanto espaciais.
- **Roteirização e programação**: há restrições de precedência entre tarefas e/ou restrições de janela de tempo.

Neste trabalho utilizou-se a roteirização pura.

2.1 O PROBLEMA DO CAIXEIRO-VIAJANTE (PCV)

O mais conhecido dos problemas de otimização combinatória é o Problema do Caixeiro-Viajante (PCV). De acordo com Zamboni (1997) a primeira utilização do termo “Problema do Caixeiro-Viajante” no meio matemático foi entre 1931 e 1932. Porém a essência do Problema do Caixeiro-Viajante aparece no livro “*Der Handlungsreisende, wie er sein sol und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein. Von einem alten Commis-Voyageur”*, publicado na Alemanha em 1832.

Dantzig, Fulkerson e Johnson (1954) propõem uma “Solução de Larga Escala para O Problema do Caixeiro-Viajante” no Jornal da Sociedade de Pesquisa Operacional da América.

O Problema do Caixeiro-Viajante consiste em estabelecer uma rota única, de distância mínima, que passe em cada nó de um grafo uma única vez, retornando ao nó inicial ao final do percurso.

Solucionar o PCV significa encontrar uma permutação dos vértices com um custo mínimo. Um problema com N cidades tem seu espaço de busca definido pela permutação $(n-1)!$, sendo n o número de nós. O PCV pertence à classe de problemas *NP-Hard*, ou seja, não existem algoritmos polinomiais capazes de resolvê-lo. Devido ao esforço computacional, torna-se inviável resolvê-los através de métodos exatos.

O Problema do Caixeiro-Viajante possui muitas variações sendo que algumas possuem algoritmos de aproximações que fornecem resultados próximos do ótimo. Essas variações podem ser classificadas em relação a:

- **Simetria**: Quando a distancia do nó “A” ao nó “B” é igual ao a distância do nó “B” ao nó “A”, o PCV é dito simétrico, caso contrário é dito assimétrico.
- **Completude**: O PCV é dito completo quando existe um caminho direto entre todos os nós, caso contrário é dito não completo.

Suponha que o grado G é completo e com peso nas arestas. O PCV restrito ao conjunto de instâncias em que G é completo e o custo de suas arestas satisfaz a desigualdade triangular, isto é, $c_{ij} + c_{jk} \geq c_{ik}$ para quaisquer três vértices i, j, k, são conhecidos como métricos.

Por exemplo, considerando os vértices de um grafo como pontos do plano, o custo de se viajar de um vértice a outro é a distância euclidiana:

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

As restrições usadas nesse trabalho são tais que o grafo seja simétrico, completo e que a desigualdade triangular seja satisfeita. E a métrica utilizada é a distância euclidiana.

2.1.1 Formulação Matemática para o Problema do Caixeiro-Viajante

Seja o Grafo $G(N, A)$ onde N representa o conjunto dos vértices e A o conjunto das arestas. Seja a matriz simétrica, com os custos mínimos entre os nós, $C = [c_{ij}]$, e considerando $c_{ii} = +\infty$, $\forall i \in N$. A matriz das variáveis de decisão do problema é $X = [x_{ij}]$, onde $x_{ij} = \begin{cases} 1, & \text{se a aresta } a_{ij} \in \text{rota} \\ 0, & \text{se a aresta } a_{ij} \notin \text{rota} \end{cases}$.

A modelagem clássica para o PCV em problema de programação inteira bivalente de acordo com Dantzig, Fulkerson e Johnson (1954), é dada por:

$$Minimizar\ x_0 = \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} \cdot x_{ij}$$

$Sujeito\ a$:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in N \qquad (i)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in N \qquad (ii)$$

$$\sum_{i,j \in S}^{n} x_{ij} \leq |S| - 1 \qquad \forall S \subset N \qquad (iii)$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in N \qquad (iv)$$

Onde a variável binária x_{ij} assume valor igual a 1 se o arco $(i, j) \in A$ for parte integrante da solução, e 0 caso contrário. S é um subgrafo de G em que $|S|$ representa o número de vértices desse grafo. As restrições i e ii garantem que cada vértice seja visitado apenas uma vez, mas essas duas restrições não eliminam os ciclos isolados, chamados de *subtours*. A restrição iii garante a eliminação dos ciclos isolados.

Pode-se concluir através dessa formulação que solucionar o PCV significa encontrar uma permutação dos vértices com um custo mínimo. Um problema com N cidades tem o número de possíveis rotas que podem ser percorridas dado por:

$$P = \frac{(N-1)!}{2}$$

Por exemplo, para um problema com 10 cidades, tem-se um total de 1.814.400 permutações. Devido à grande dimensão de P, mesmo para poucas cidades, a busca exaustiva seria inviável devido ao tempo computacional.

3. OTIMIZAÇÃO, HEURISTICAS E METAHEURISTICAS

3.1 OTIMIZAÇÃO

De forma natural, formulações matemáticas são propostas para encontrar-se soluções de problemas encontrados na natureza. A teoria da otimização pode ser definida como a tarefa de determinar qual a solução ótima, ou seja, melhor solução existente para um determinado problema. A Matemática é a responsável, na maioria dos casos, pela formulação dos métodos de busca dessas soluções.

Basicamente um problema de otimização obedece a um modelo básico de raciocínio, que é: minimizar ou maximizar uma função objetivo $f(x)$, obedecendo-se um conjunto de restrições.

Alguns problemas de otimização envolvem modelos lineares, onde a variáveis são contínuas e apresentam um comportamento linear (GREIG, 1980). Outros problemas são formulados por modelos não-lineares, e tem como característica exibir qualquer tipo de não-linearidade ou não continuidade, tanto em sua função objetivo quanto em suas restrições. Naturalmente esses problemas apresentam uma maior complexidade e dificuldade de resolução (CLERC; KENNEDY, 2002).

As soluções encontradas podem ser globais ou locais. Caso a solução seja a melhor entre todas as soluções existentes encontradas ela é dita solução ótima global. No entanto se a solução encontrada for a melhor em apenas um conjunto de restrições, esta é chamada de solução ótima local.

3.1.1 Otimização Local e Global

Considere-se uma função $f: A \subset S \rightarrow S \subseteq R^n$ de variável x, e onde S denota o espaço de busca. Pode-se definir como ponto de *mínimo local* de uma função f(x) da seguinte forma:

$f(x^*) \leq f(x), \forall x \in A$.

Se um problema não possui restrições temos que $S = R^n$, e A é um subconjunto de S. O espaço de busca S pode conter varias sub-regiões A_i tais que $A_i \cap A_j = \emptyset$ quando $i \neq j$, sendo que pontos definidos nessas sub-regiões são únicos, isto é, $x^*_{Ai} \neq x^*_{Aj}$·, onde qualquer ponto x^*_{Ai} pode ser considerado mínimo de A_i assim

como qualquer ponto x_{Aj}^* pode ser considerado mínimo de A_j. Não há restrições para os valores mínimos da função, portanto pode-se ter $f(x_{Ai}^*) = f(x_{Aj}^*)$.

Os algoritmos de otimização, em grande maioria das vezes, necessitam de um ponto inicial $x_0 \in S$. Para que o algoritmo de otimização local seja eficaz, ele deve garantir que é capaz de encontrar um mínimo local x_{Ai}^* para um conjunto A, se $x_0 \in A$.

O *mínimo global* x^* de uma função $f(x)$, é definido da seguinte forma:

$f(x^*) \leq f(x), \forall x \in S$.

A otimização global também deve começar com a escolha de um ponto inicial $x_0 \in S$. Caso o problema não possua restrições comumente usamos $S = R^n$, onde n é a dimensão de x.

A Figura 3 a seguir apresenta a diferença entre um mínimo local e um mínimo global.

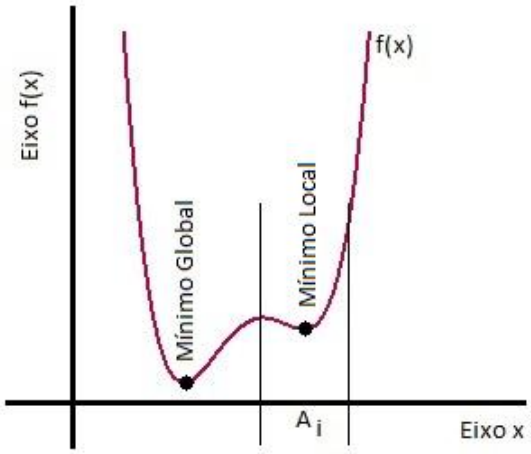

FIGURA 3: EXEMPLO DE FUNÇÃO COM MÍNIMO LOCAL E MÍMIMO GLOBAL

Fonte: O Autor (2014)

De acordo com Dennis e Schnabel (1983), os algoritmos de otimização global são capazes de localizar, independentemente da posição atual de x_0, o mínimo (local) de $A \subset S$, e consistem em duas etapas composta por passos globais e passos locais, onde a etapa de passos globais é responsável por garantir que o algoritmo passa para a região A_i, e a etapa de passos locais é responsável por encontrar um mínimo em A_i. Esses algoritmos são conhecidos como globalmente convergentes, podendo ser usados para encontrar mínimos locais ou globais. Porém não é possível assegurar a localização de um mínimo global.

3.1.2 Otimização Combinatória

Os problemas de otimização se dividem em duas categorias: contínuos e discretos. Nos problemas de otimização contínua as soluções são codificadas com variáveis de valores reais e no caso de otimização discreta as soluções são codificadas por variáveis discretas (BLUM; ROLI, 2003). A classe dos problemas de Otimização Combinatória – *Combinatorial Optimization* (CO) – faz parte categoria de otimização discreta.

Assim como todo problema de Otimização, os problemas de otimização combinatória possuem uma função a ser otimizada. De acordo com Papadimitriou e Steiglitz (1982), a solução de um problema de otimização combinatória é geralmente um número inteiro, um subconjunto, uma permutação ou a estrutura de um grafo. Esta solução pode ser um ótimo global ou ótimo local, dependendo da sua localização no espaço de busca.

Blum e Roli (2003), definem um Problema de Otimização Combinatória, $COP = (S, f)$, com as seguintes características:

- Um conjunto de variáveis $X = \{x_1, \ldots, x_n\}$;
- Os domínios das variáveis $D_1, \ldots, D_n$;
- Uma função objetivo f a ser minimizada ou maximizada, onde $f: D_1 \times \ldots \times D_n$;
- Restrições entre as variáveis.

Definindo-se o conjunto de todas as possíveis atribuições viáveis como:

$$S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} | v_i \in D_i\}$$

onde S é o espaço de busca, onde cada elemento é uma possível solução e s satisfaz todas as restrições. Resolver o COP significa encontrar um elemento $s^* \in S$, satisfazendo a condição de mínimo global. Esta solução pode também ser chamada de solução ótima global de (S, f). E o conjunto $S^* \subseteq S$ é chamado de conjunto de soluções ótimas globais.

Uma das principais diferenças entre os problemas de Otimização combinatória e os demais problemas de otimização é em relação ao domínio das variáveis, que representa o conjunto dos possíveis valores discretos que a variável pode assumir.

3.2 HEURÍSTICAS

Reeves (REEVES, 1995), define uma heurística como uma técnica que, com um custo computacional razoável, busca boas soluções (quase ótimas), porém não é capaz de garantir que as mesmas sejam ótimas ou admissíveis. Existem casos em que não se podem nem determinar o quão próximo a solução admissível está da solução ótima.

O foco da heurística está na busca contínua com a mínima utilização de recursos computacionais e na sua flexibilidade, já que a mesma não pode garantir a descoberta da solução ótima.

As técnicas heurísticas permitem resolver de maneira aproximada o Problema do Caixeiro-Viajante. De acordo com Bodin (1983), existem três grandes procedimentos para resolver o PCV:

1. Procedimentos de construção de rotas, que constroem rotas ótimas ou quase ótimas.
2. Procedimentos de melhorias de rotas, que efetuam melhorias em rotas já existentes.
3. Procedimentos compostos, que constroem uma rota inicial com auxilio dos procedimentos de construção e utilizam os procedimentos de melhorias para obter um resultado mais eficiente.

Recentemente, surgiram as técnicas conhecidas por Metaheurísticas, destacam-se os Algoritmos Genéticos, Colônia de Formigas, *Simulated Annealing*, GRASP e Busca Tabu. Essas metaheurísticas serão abordados no próximo capitulo.

3.2.1 Heurística de Construção de Rota

Para o problema do caixeiro-viajante, as heurísticas de construção de rotas geram um circuito viável a partir de um conjunto de inicial de vértices, e utilizando algum critério de escolha modificam esse conjunto a cada iteração.

Segundo Bodin *et al* (1983), as rotas podem ser inicialmente construídas utilizando as seguintes técnicas de construção de rotas:

- Procedimento do vizinho mais próximo
- Inserção do mais próximo

- Inserção do mais distante
- Inserção mais rápida

Existem outros procedimentos de construção de rotas descritos em Bodin *et al* (1983), tais como Inserção do Mais Barato, Inserção Arbitrária, Cobertura Convexa, Inserção do Maior Ângulo.

Utilizando Heurísticas de melhoria de rotas, eventualmente é possível melhorá-las. Dentre as técnicas de melhorias de rotas podem ser citadas as técnicas de troca de arcos, mais conhecidas por 2-opt e 3-opt (LIN, KERNGIHAN, 1973).

3.2.1.1 Vizinho Mais Próximo

É a técnica mais intuitiva das heurísticas e trabalha da seguinte forma: o ciclo inicia com um nó v_i, e a partir deste, encontra o próximo nóv_k, de forma que a distância entre a esses dois nós seja mínima. Após encontrar o nó v_k, o algoritmo insere o mesmo ao final do ciclo e repete a operação até que todos os nós pertençam à solução. Então o ciclo é fechado ligando o nó inicial ao nó final. Não é permitido visitar mais de um nó duas vezes ou modificar sua escolha, assim após um nó ser inserido em uma determinada posição da rota, o mesmo não poderá sofrer modificação em sua posição(BODIN, 1983).

O tempo de computação envolvido ao Vizinho mais próximo é da ordem n^2.

O pseudocódigo da heurística do Vizinho mais próximo é representado no algoritmo 1:

P1: Escolha um nó Inicial;
P2: Encontre o nó mais próximo do ultimo nó adicionado e adicione-o na rota;
P3: Repita o passo 2 até que a rota contenho todos os nós e retorne ao primeiro.

ALGORITMO 1 - Algoritmo do Vizinho mais próximo

3.2.1.2 Inserção do Mais Próximo

O algoritmo da inserção do mais próximo é uma heurística que possui um processo onde três níveis de decisão estão envolvidos: a escolha do vértice a ser inserido, a posição de inserção e a decisão de um ciclo inicial (GOLDBARG, 2005).

O algoritmo da inserção do mais próximo também possui tempo computacional de ordem n^2.

O pseudocódigo do algoritmo da heurística da inserção do vizinho mais próximo é representado a seguir no algoritmo 2:

P1: Inicie com um sub-grafo contendo apenas o nó i;
P2: Encontre um nó k tal que c_{ik} seja mínima e forme a rota $i - k - i$;
P3: dada a sub-rota, encontre o nó k, não pertencente à sub-rota, mais próximo de qualquer nó da sub-rota;
P4: Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$. Insira k entre i e j;
P5: Volte ao passo 3 até formar um circuito Hamiltoniano.

ALGORITMO 2 – Algoritmo da Inserção do mais próximo

3.2.1.3 Inserção do Mais Distante

Esse algoritmo é semelhante ao da heurística Inserção do Mais Próximo, difere apenas no passo 2, quando então se escolhe a cidade k não pertencente ao ciclo, mais distante de qualquer cidade do ciclo. Também, no passo 3, o nó k não pertencente a sub-rota deve ser mais distante de qualquer nó e não o mais próximo. O tempo de computação envolvido ao algoritmo de Inserção do Mais Distante também é da ordem n^2. No algoritmo 3 está representado o pseudocódigo da heurística da Inserção do mais distante:

P1: Inicie com um sub-grafo contendo apenas o nó i;
P2: Encontre o nó k tal que cik seja máxima e forme a rota $i - k - i$;
P3: Dada a sub-rota, encontre o nó k não pertencente a sub-rota mais distante de qualquer nó da sub-rota;
P4: Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$. Insira k entre i e j;
P5: Volte ao passo 3 até formar um circuito Hamiltoniano.

ALGORITMO 3 - Algoritmo da Inserção do mais distante

3.2.1.4 Inserção mais rápida

O pseudocódigo do Algoritmo 4 representa a heurística da inserção mais Rápida. Também possui tempo de computação da ordem n^2.

P1: Inicie com um sub-grafo contendo apenas o nó i;
P2: Encontre o nó k tal que cik seja máxima e forme a rota $i - k - i$;
P3: Dada a sub-rota, encontre o nó k não pertencente a sub-rota mais distante de qualquer nó da sub-rota;
P4: Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$. Insira k entre i e j;
P5: Volte ao passo 3 até formar um circuito Hamiltoniano.

ALGORITMO 4 - Algoritmo da Inserção mais Rápida

3.2.2 Heurística de Melhoria de Rota

As heurísticas de melhoria de rotas baseiam-se em modificações simples no circuito, e oferecem bons resultados. Dado um circuito hamiltoniano, essas heurísticas fazem trocas para que seu comprimento seja reduzido, ate que seja impossível reduzi-lo mais, encontrando um circuito localmente ótimo.

Essas heurísticas são usadas como parte de outras técnicas usadas no PCV,

As heurísticas 2-opt e 3-opt surgiram na década de 60 e consistem em permutar arcos de uma rota inicial factível, na busca de uma rota de menor custo.

Na heurística dois 2-opt, dois arcos são desligados e substituídos por outros dois de modo que a distância total na nova rota seja menor que na rota inicial. Na heurística 3-opt, três arcos são permutados. Geralmente estas heurísticas encontram um ótimo local, e são considerados métodos eficientes para resolver o PCV. Posteriormente foi criada a heurística k-opt, onde são permutados k arcos (k≥3), mais forte que a 2-opt e a 3-opt. A medida que k aumenta, as soluções melhoram, porém quando k≥4, esta técnica tem custo computacional muito alto, tornando-a quase impraticável.

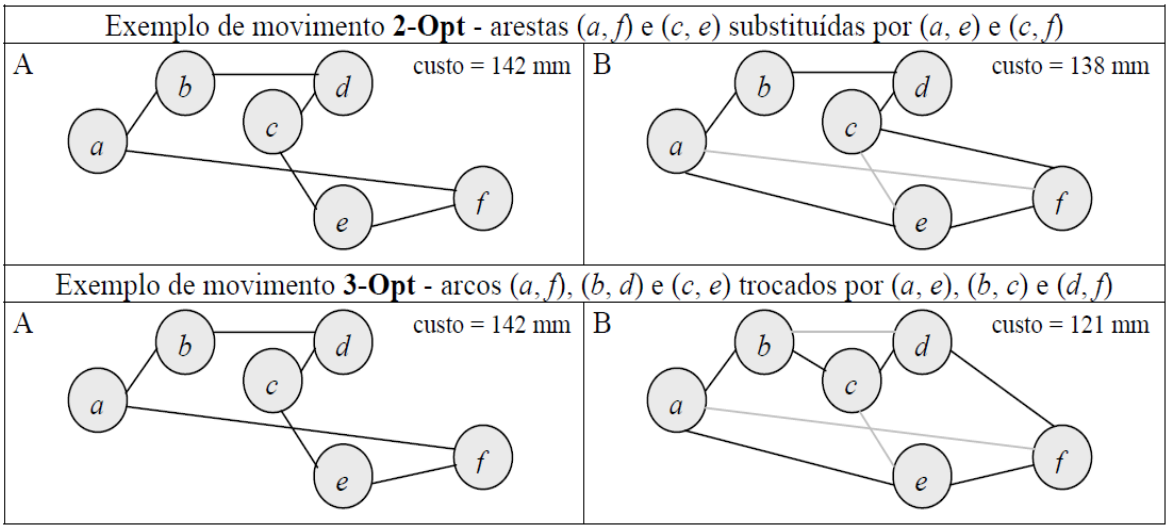

FIGURA 4 – EXEMPLOS DE SUBSTITUIÇÂO 2-OPT E 3-OPT

Fonte: (GANHOTO, 2004)

O algoritmo 5 representa os passos das heurísticas 2-opt e 3-opt:

P1: Obtenha uma rota inicial factível, utilizando algum procedimento de formação de rotas;

P2: No caso 2-opt, desligar 2 arcos e no caso 3-opt desligar 3 arcos, reconectando os nós por meios de arcos diferentes daqueles que foram desconectados e que determinem uma nova rota factível. Se o comprimento da nova rota for menor que a rota anterior, troque a rota atual pela rota nova.

P3: Repetir o passo 2 até que nenhuma melhoria possa ser alcançada.

ALGORITMO 5 - Algoritmo de Melhoria 2-opt e 3-opt

O algoritmo de busca local desenvolvido por Lin e Kernighan (1973) conhecido como LK, é atualmente a mais conhecida e utilizada busca local para o problema do caixeiro-viajante simétrico. Esta heurística se baseia no método k-opt, porém com a diferença de que o valor de k passa a ser variável. E as trocas de arcos são realizadas segundo um critério de ganho que restringe o tamanho da vizinhança de busca. Este algoritmo determina qual a melhor troca e também qual o melhor k para aquela iteração. Além de ter um custo computacional relativamente baixo.

O algoritmo busca uma sequência de arcos $\{x_1, x_2, \dots, x_k\}$ que aquando trocadas pelos arcos $\{y_1, y_2, \dots, y_k\}$ retornam um caminho factível e de menor custo. O algoritmo LK inicia construindo um caminho aleatório e sorteando um nó n_1 base pelo

qual começaram a trocas. Em seguida é escolhida uma aresta que contem o nó inicial $x_1 = (n_1, n_2)$ e uma aresta de troca que parta de n_2, e gere um ganho positivo $y_1 = (n_2, n_3)$. O nó n_3 é chamado vizinho promissor de n_2.

Escolhida a primeira aresta de troca, começa então o processo iterativo com $i = 2$, onde a cada passo uma aresta é escolhida contendo o último nó escolhido na iteração anterior $x_i = (n_{2i-1}, n_{2i})$, e escolhe-se uma aresta partindo de n_{2i} tal que algumas condições sejam satisfeitas:

i) Se a aresta (n_1, n_{2i}) é criada, então um caminho completo é formado;

ii) A aresta y_i é uma aresta não utilizada contendo o nó n_{2i};

iii) Para garantir a disjunção entre x_i e y_i, x_i não pode ser uma aresta já escolhida para o conjunto y, e y_i não pode ser uma aresta já escolhida pelo conjunto x;

iv) O ganho deve ser positivo;

v) A aresta y_i deve permitir a quebra de x_{i+1} para possibilitar que na iteração seguinte existam trocas factíveis;

vi) Antes da decisão final da escolha de y_i é verificado se, ao fechar n_{2i} com n_1, gera-se um caminho com custo menor do que o original.

Terminada a construção do conjunto de arestas originais (x) e o conjunto de arestas de troca (y), o novo caminho é construído e os passos são executados novamente até que não ocorra mais nenhum ganho.

A literatura contem relatos de diferentes implementações do LK, com grande variação de comportamento (JOHNSON;MCGEOCH, 2001).

4. METAHEURÍSTICAS

As metaheurísticas têm como objetivo encontrar, assim como as heurísticas, uma solução com reduzido esforço computacional e com flexibilidade controlada. E tem como característica principal a capacidade de escapar de ótimos locais (CHAVES, 2003).

Algumas das características que diferenciam as metaheurísticas são (POLTOSI, 2006):

- Definição da escolha e representação da solução inicial;
- Definição de uma vizinhança $N(t)$, de uma solução t;
- Critério de seleção de uma solução vizinha dentro de $N(t)$;
- Condição de término.

Perante estas características as metaheurísticas agruparam-se em dois grupos, busca local e busca populacional, segundo seus critérios utilizados na busca por soluções.

A seguir são descritas algumas metaheurísticas aplicadas ao PCV.

4.1 SIMULATED ANNEALING

No inicio da década de 80, KIRKPATRICK, GELATT & VECCHI (1983) apresentaram o conceito de recozimento (a*nnealing*) em problema de otimização combinatória. A técnica *Simulated Annealing* tem sua origem na mecânica estatística. E computacionalmente pode ser visto como um processo estocástico de determinação de uma organização dos átomos, que a partir do resfriamento gradativo de um material, com alta temperatura inicial leva o material a estados mínimos de energia. Esses estados são caracterizados por uma perfeição estrutural do material resfriado que não seria obtida caso o resfriamento não fosse gradativo.

A ideia proposta ao utilizar o *simulated annealing* é a seguinte:

Identificar a função de energia do sistema como a função objetivo que se quer otimizar, neste caso minimizar, e os átomos do sistema são associados as variáveis do problema;

Para a simulação a uma temperatura fixa *T* consiste em causar um pequeno deslocamento no átomo, calculando a variação ΔE da energia do sistema. Se $\Delta E \leq 0$ o

deslocamento é incorporado ao estado do sistema, caso contrário, a aceitação ou não do deslocamento passa a ser uma decisão probabilística.

Para cada temperatura de uma sequência de temperaturas decrescentes realiza-se a simulação descrita. Espera-se que no final do processo o sistema estacione no estado de energia mínima global.

O método do *simulated annealing* permite a aceitação de estados intermediários em que o valor da função objetivo, a que se que minimizar, aumenta. O método admite soluções intermediárias "piores" na esperança fugir de ótimos locais e encontrar ótimos globais.

A redução da temperatura foi descrita originalmente como a função $\mathrm{r(k)} = \alpha^{\mathrm{k+1}}\mathrm{T_0}$, para todo $\mathrm{k} \geq 0$ e algum $0 < \alpha < 1$.

A probabilidade do algoritmo aceitar uma solução factível *j* a partir de uma solução *i*, é dada por:

$$\mathrm{Prob\{j\ vir\ depois\ de\ i\}} = \begin{cases} 1 & se\ \Delta \leq 0 \\ e^{-\frac{\Delta}{T}} & se\ \Delta > 0 \end{cases} \qquad (2)$$

onde $\Delta = \mathrm{f(j)} - \mathrm{f(i)}$.

A medida que o valor de T decresce torna-se cada vez mais improvável a aceitação de soluções com maior valor na função objetivo.

De acordo com Barbosa (1989) e Mitra (1986), a fim de garantir a convergência para um mínimo local, deve-se selecionar $\mathrm{T_0}$ e $\mathrm{r(k)}$ de forma que:

$$\mathrm{T_0} = \frac{\varepsilon}{\log(1 + \mathrm{k_0})} \qquad (3)$$

$$\mathrm{r(k)} = \frac{\varepsilon}{\log(2 + \mathrm{k_0} + \mathrm{k})} \qquad (4)$$

para $k \geq 0$, onde k_0 é qualquer parâmetro satisfazendo $1 \leq k_0 \leq k$ e ε é uma constante que depende das características da cadeia de Markov associada ao Processo de *Simulated Annealing*.

Como a verificação da verdadeira condição de equilíbrio da temperatura T é difícil, então introduz-se um parâmetro H, que fixa o número máximo de iterações para uso da temperatura T, antes que a mesma seja adaptada.

O Algoritmo 6, segundo Barbosa (1989) apresenta os passos da metaheurística *Simulated Annealing*:

Seja $x_0 \in D$ a solução inicial e T_0 a temperatura inicial;

$k = 0$; $x_k = x_0$; $T_k = T_0$

Enquanto $T_k \geq T_{min}$ faça

Inicio

Enquanto a temperatura $T_k$não atinge o equilíbrio **faça**

Inicio

Selecione aleatoriamente um ponto $x' \in A(x_k) \subset D$

$\Delta = \mathbf{f}(\mathbf{x}') - \mathbf{f}(\mathbf{x})$

Se $\Delta \leq 0$ **então**

$x_k = x'$

Senão

$x_k = x'$ com probabilidade $e^{-\frac{\Delta}{T_k}}$

Fim

$T_{k+1} = r(k)$; $x_{k+1} = x_k$; $k = k + 1$

Fim

Retorne a solução x_k.

ALGORITMO 6 - Algoritmo *Simulated Annealing*

4.2 ALGORITMOS GENÉTICOS

Os Algoritmos Genéticos (AG) foram introduzidos por Jonh Holland em 1975 e constituem um método de otimização inesperado no processo Darwiniano de seleção natural dos seres vivos e reprodução genética. Pertence a uma classe de paradigmas e técnicas computacionais inspiradas na evolução natural, denominada Computação Evolucionista.

De acordo com Grefenstette (1986) é um processo iterativo que mantem uma população estruturas, chamadas de indivíduos, que representam as possíveis soluções para um determinado problema. A cada iteração, os indivíduos passam por uma avaliação que verifica sua capacidade de oferecer uma solução satisfatória para o problema. Esta avaliação é feita através da função de aptidão ou função *fitness*. De acordo com essa avaliação, alguns indivíduos são selecionados para passar por um processo de reprodução. Aplicando sobre os indivíduos selecionados uma série de operadores genéticos é gerada uma nova população. Supõem-se que a cada iteração

a população fica mais apta para solucionar o problema, e após um determinado número de iterações, ou gerações, de acordo com algum critério de parada, o indivíduo mais apto é uma possível solução para o problema. Os AG são uma ferramenta muito versátil e robusta, pois apesar de não garantir uma solução ótima, apresentam soluções quase ótimas para problemas complexos. Os AG permitem uma melhor exploração do espaço de busca pois evitam ser atraídos por ótimos locais.

As principais características dos AG são:

- Emprego de uma população de indivíduos que podem ser de tamanho fixo ou variável;
- Não trabalham diretamente com as possíveis soluções do problema, chamadas de fenótipos, e sim com uma codificação das mesmas chamadas de genótipos;
- Empregam regras probabilísticas ou estocásticas.
- Não exigem maiores informações sobre a função a otimizar.

O pseudocódigo que representa os AG, segundo Barbosa(1997) é apresentado a seguir:

```
Inicie a população
   Avalie os indivíduos da população
   Repita
      Selecione indivíduos para reprodução
      Aplique operadores de recombinação e mutação
      Avalie indivíduos da população
      Selecione indivíduos para sobreviver
   Até critério de parada ser satisfeito
Fim
```

ALGORITMO 7 - Algoritmo Genético

4.2.1 Representação e Codificação

A primeira etapa para resolver um problema utilizando algoritmos genéticos é a representação e codificação dos elementos do espaço de busca. Chama-se de fenótipo os elementos do espaço de busca, e o código que o representa chama-se de

genótipo. A codificação é a função que associa os genótipos aos fenótipos. A representação mais comum é a binária, onde o alfabeto é composto pelos símbolos 0 e 1. No sistema binário, a cadeia de *strings* “10011010”, de comprimento l, representa uma possível solução para o problema. Neste caso temos que o conjunto dos genótipos é formado por todos os números binários de “00000000” a “11111111”, com um total de $2^8 = 256$ elementos. Portanto a codificação associa a cada uma dessas cadeias binarias uma solução. Um elemento do espaço de busca composto pelas variáveis v_i, de diferentes tipos pode ser codificado como uma cadeia da forma “$\underbrace{1100101}_{v_1}\underbrace{001001}_{v_2}\ldots\underbrace{10111101}_{v_n}$” também é denominado cromossomo. Cada cromossomo é composto por subcadeias, denominadas genes. Um genótipo pode ser representado por um ou mais cromossomos.(NUNES, 1998).

4.2.2 Função de Aptidão

A função de aptidão esta diretamente ligada à função objetivo, e reflete a qualidade de um elemento solucionar um dado problema. No caso do Problema do Caixeiro-Viajante a função de aptidão pode ser a função que fornece a distancia total percorrida pelo viajante.

Com o processo natural de evolução os indivíduos passam a tem aptidões (*fitness*) cada vez mais semelhantes entre si, podendo ser necessário aumentar a pressão de seleção pela adoção de alguma estratégia, como por exemplo a composição da função objetivo com alguma função escolhida convenientemente.

4.2.3 Seleção

O processo de seleção baseia-se no principio da sobrevivência do melhor indivíduo, onde os cromossomos com melhor aptidão tem maior probabilidade de serem selecionados para participar do processo de reprodução.

No processo de seleção proporcional à aptidão, a probabilidade p_i do indivíduo a_i ser selecionado para reprodução é o cálculo proporcional ao seu valor da função de aptidão, calculado pela fórmula:

$$p_i = \frac{f(a_i)}{\sum_{j=1}^{n} f(a_j)} \tag{5}$$

onde f é a função de aptidão e n o tamanho da população.

Após definida a forma de quantificação da probabilidade de sobrevivência, um dos métodos de seleção dos indivíduos é o conhecido “método da roleta” onde a probabilidade de cada indivíduo ser selecionado é proporcional ao seu *fitness*.

4.2.4 Reprodução

O processo de seleção não introduz novos indivíduos na população. Na etapa de reprodução o algoritmo tenta criar novas e melhores soluções, isto é, indivíduos mais aptos. Existem duas classes de algoritmos genéticos quanto a forma que novos indivíduos são inseridos na população (BARBOSA, 1997).

A primeira classe é chamada de Algoritmo Genético Generacional, onde toda a população é substituída pelos novos indivíduos criados após o processo de seleção e aplicação dos operadores genéticos.

O pseudocódigo do AG generacional é representado a seguir:

```
Inicie a população P de alguma forma
    Avalie os indivíduos da população P
    Repita
        Repita
            Selecione indivíduos da população P
            Aplique os operadores genéticos
            Insira os novos indivíduos em P'
        Até que a população P' esteja completa
        Avalie os indivíduos da população P
        P ← P′
    Até que algum critério de parada seja satisfeito
Fim
```

ALGORITMO 8 - Algoritmo Genético Generacional

Neste processo corre-se o risco de substituir bons indivíduos da população, a fim de evitar isto, pode-se utilizar o processo elitista que repassa a copia de alguns dos melhores indivíduos para a geração seguinte.

A segunda classe é conhecida como Algoritmos Genéticos “*steady-state*” que cria um individua de cada vez e pode ou não repassar esse indivíduo para a geração seguinte. Normalmente ele é transmitido se seu valor de *fitness* for melhor do que o pior valor de *fitness* da população antiga. O pseudocódigo a seguir representa os AG “*steady-state*”:

```
Inicie a população de alguma forma
    Avalie os indivíduos da população P
    Ordene a população de acordo com seu fitness
    Repita
        Selecione indivíduos da população P
        Aplique operadores genéticos
        Selecione um individuo f para sobreviver
        Se f é melhor que o pior elemento de P então
            Remova um individuo da população
            Insira f em P de acordo com seu “ranking”
    Até que algum critério de parada seja satisfeito
Fim
```

ALGORITMO 9 - Algoritmo Genético “steady-state”

4.2.5 Operadores Genéticos

O principio básico dos operadores genéticos é transformar a população através de sucessivas gerações para obter um resultado satisfatório ao final do processo.

4.2.5.1 Operador de Recombinação

Os operadores de recombinação atuam sobre o genótipo dos indivíduos selecionados. Promovendo o intercambio de informações genéticas dos elementos “pais”, gerando elementos “filhos”. Este operador também é conhecido como crossover, e pode ser utilizado de varias maneiras, onde as mais empregadas são:

Um-ponto: um ponto de cruzamento é escolhido aleatoriamente e a partir dele as informações genéticas dos pais são trocadas, conforme o exemplo:

$$\mathrm{p}_1: 1111\ \ 00000 \quad \mathrm{f}_1: 1111\ \ 11100$$
$$\mathrm{p}_2: 1110\ \ 11100 \quad \mathrm{f}_2: 1110\ \ 00000$$

onde os elementos p_1e p_2 são os pais e os elementos f_1 e f_2 são os filhos

Multi-pontos: uma generalização da troca de material genético onde mais de um ponto podem ser selecionados

Uniforme: Não determina pontos de cruzamento, mas determina através de um parâmetro global, qual a probabilidade de cada variável ser herdada do “pai”.

4.2.5.2 Operador de Mutação

O operador de mutação é necessário para a introdução e manutenção de variabilidade genética na população. De acordo com Barboza (2005) seleciona-se uma posição de um cromossomo e altera-se aleatoriamente o valor do gene para outro alelo possível. A taxa de mutação deve se relativamente pequena, pois uma taxa muito alta pode tornar a busca essencialmente aleatória.

Os principais tipos de mutação são:

- **Mutação Simples:** Considerando o alfabeto binário, uma posição do cromossomo é sorteada e o bit correspondente é invertido, isto é, se o bit for 0 passa a ser 1 e vice-versa.
- **Mutação por Troca (*Swap*):** Consiste na troca aleatória de posição entre dois genes. Por exemplo, temos o indivíduo {1,2,3,4,5} e que após a mutação torna-se {1,4,3,2,5}.

4.2.6 Algoritmo Genético para o PCV

Existem diversas abordagens para o PCV utilizando os algoritmos genéticos, segundo Whitley et al (1989) e que diferenciam nos parâmetros, na representação das soluções, na escolha dos indivíduos para reprodução e na definição dos operadores genéticos.

Segundo Bezerra (1995), uma solução para o Problema do Caixeiro-Viajante composto por n cidades pode ser considerada como sendo o conjunto

$X = \{x_1, x_2, \dots, x_n\}$. Sendo conhecido o custo da viagem entre cada par de cidades, dado por $w(x_i, x_j), \forall x_i, x_j \in X$, e $w(x_i, x_j) > 0$, pode-se definir a estrutura do cromossomo, a função de aptidão e os operadores de *crossover* e mutação de forma a seguir.

4.2.6.1 Estrutura do cromossomo

As principais representações são: ordinal, por caminho e por adjacência.

- **Representação Ordinal:** A solução é representada com uma lista de n cidades, onde o i-ésimo elemento da lista é um número entre 1 e (n-i+1). A lista $\mathbf{L}$, ordenada de 1 até n serve de referência a construção dessa representação, e o cromossomo é um vetor de posicionamento nessa lista. Exemplo: $\mathrm{L} = \{1,2,3,4,5\}$
- Considere o cromossomo $\mathrm{C}_1 = \{5,1,3,2,1\}$, o primeiro elemento do cromossomo C_1 é 5, e corresponde ao quinto elemento da lista $\mathbf{L}$, que é 5 (Rota: 5), em seguida retira-se esse elemento da lista $\mathbf{L}$. O próximo elemento de C_1 é 1 que corresponde ao primeiro elemento da lista, que é 1 (Rota: $\{5, 1\}$) e retira-se o 1 da lista $\mathbf{L}$. O próximo elemento de C_1é 3, correspondente ao 4 (Rota:$\{5, 1, 4\}$). Seguindo esse raciocínio chegamos à rota final $\mathrm{R}_1 = \{5, 1, 4, 3, 2, 5\}$, onde repete-se o primeiro elemento para fechar o ciclo.

  Portanto $\mathrm{C}_1 = \{5,1,3,2,1\} \rightarrow \mathrm{R}_1 = \{5, 1, 4, 3, 2, 5\}$.
- **Representação por Caminhos:** O cromossomo é formado pela sequência dos nós da solução, isto é, o cromossomo $\mathrm{C}_1 = \{1, 2, 3, 4, 5\}$ representa a rota $\mathrm{R}_1 = \{1, 2, 3, 4, 5\}$.
- **Representação por Adjacência:** De acordo com Mayerle (1994), cada cromossomo representa um circuito hamiltoniano, onde cada vértice possui um sucessor. Portanto da forma $\mathrm{R_i} = \{\mathrm{s_{i1}}, \mathrm{s_{i2}}, \dots, \mathrm{s_{in}}\}$, onde o gene $\mathrm{s_{ij}}$ é o vértice sucessor do vértice $\mathrm{x_j}$ no i-ésimo circuito.

  Por exemplo: Um cromossomo $C_1 = \{5, 3, 4, 1, 2\}$ representa os seguintes arcos da rota $1 \rightarrow 5, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 1, 5 \rightarrow 2$. Esse cromossomo representa a seguinte solução $R_1 = \{1,5,2,3,4\}$.

4.2.6.2 Função de Aptidão

De acordo com Mayerle (1994) Para os problemas de minimização, quanto menor for o custo total do circuito, melhor será a aptidão do cromossomo correspondente e maiores as suas chances do indivíduo sobreviver e se reproduzir.

4.2.6.3 Processo de Seleção

De acordo com Barboza (2005), a seleção elitista consiste em copiar ou reproduzir os melhores indivíduos da geração atual para as próximas gerações, sem destruir esses cromossomos nas etapas de mutação e recombinação.

A escolha de um indivíduo de uma a população com m cromossomos, dispostos em ordem crescente dos seus custos, é feita considerando a distribuição de probabilidade proporcional ao índice dos cromossomos, é definida segundo Nunes (1998) pela fórmula:

$$\mathrm{Select(R)} = \left\{ \mathrm{r_i} \in \mathrm{R|j} = \mathrm{m} + 1 - \left\lceil \left(-1 + \frac{\sqrt{1 + 4.\mathrm{Rnd}(\mathrm{m}^2 + \mathrm{m})}}{2} \right) \right\rceil \right\} \qquad (6)$$

Onde:

- $\mathrm{Rnd} \in [0,1)$ é um número aleatório uniformemente distribuído;
- $\lceil \mathrm{b} \rceil$ é o menor inteiro maior do que b;
- $R = \{r_1, r_2, \ldots, r_m\}$ é o conjunto ordenado dos cromossomos, de modo que $\mathrm{C_1} \leq \mathrm{C_2} \leq \cdots \leq \mathrm{C_m}$.

4.2.6.4 Operado de Cruzamento

Os operadores de cruzamento podem ser classificados em dois tipos: os que preservam a posição absoluta das cidades e os que preservam a ordem relativa. Na primeira classe tem-se o PMX (*Partially Mapped Crossover*) proposto por Goldberg e Lingle (1985) segundo Potvin (1996). Na segunda classe tem-se o OX (*Order Crossover*) proposto por Davis (1985) de acordo com Potvin (1996).

- **Operador PMX**: Faz um crossover parcialmente mapeado utilizando dois pontos de corte e fazendo o crossover entre esses dois pontos.

Exemplo: Considere os pais P_1 e P_2

$P_1 = (6,5|4,3,2|1)$ e $P_2 = (2,3,|6,1,5|4)$

O operador troca a cadeia entre os dois pontos de corte, e define o seguinte mapeamento:

$4 \leftrightarrow 6$, $3 \leftrightarrow 1$, $2 \leftrightarrow 5$

Dessa forma as cidades repetidas dos cromossomos pais são substituídas pelas cidades mapeadas, formando os filhos F_1 e F_2:

$F_1 = (4,2|6,5,1|3)$ e $F_2 = (5,1|4,3,2|6)$

- **Operador OX**: É um crossover de dois pontos de corte, porem a cadeia entre os dois pontos de corte é preservada. E os filhos herdam a ordem de visita dos pais.

A partir do segundo ponto de corte, as cidades de um pai são copiadas na mesma ordem. Ao atingir o final do vetor continua-se no inicio do mesmo, gerando uma sequência. A seguir insere-se esta sequência no outro pai removendo-se os repetidos, a partir do segundo ponto de corte, criando os cromossomos filhos.

Exemplo:

$P_1 = (6,5|4,3|2,1)$ e $P_2 = (2,3|5,1|4,6)$

$Sequência\ 1 = (2,1,6,5,4,3)$ e $Sequência\ 2 = (4,6,2,3,5,1)$

$F_1 = (5,1|4,3|6,2)$ e $F_2 = (4,3|5,1|2,6)$

- **Operador CX**: O operador CX não utiliza pontos de corte. Executa a recombinação de modo que cada gene do descendente vem das posições correspondentes de qualquer um dos pais.

Exemplo:

$P_1 = (1,2,3,4,5,6)$ e $P_2 = (3,6,5,1,4,2)$

A partir de P_1 escolhe-se a primeira cidade e coloca-se no filho 1:

$F_1 = (1, x, x, x, x, x)$

O gene de mesma posição em P_2 tem valor 3, procurando-se este valor em P_1, verifica-se que ele se encontra no terceiro gene, e este valor é levado para a terceira posição em F_1.

$F_1 = (1, x, 3, x, x, x)$

Verifica-se que o gene de mesma posição em P_2 tem valor 5, procurando-se este valor em P_1, verifica-se que ele se encontra no quinto gene, e este valor é levado para quinta posição em F_1.

$F_1 = (1, x, 3, x, 5, x)$

Novamente verifica-se que o gene de mesma posição em P_2 tem valor 4, procurando-se este valor em P_1, verifica-se que ele se encontra no quarto gene, e este valor é levado para a quarta posição em F_1.

$F_1 = (1, x, 3,4,5, x)$

O gene que ocupa a mesma posição em P_2possui valor 1 e encontra-se na primeira posição, porem essa operação já foi realizada. Então completamos as cidades de F_1 com as cidades de P_2 e obtemos:

$F_1 = (1,6,3,4,5,2)$

De maneira análoga, obtemos o segundo filho:

$F_2 = (3,2,5,1,4,6)$

4.2.6.5 Operador de Mutação

O Algoritmo genético pode convergir rapidamente para uma região do espaço de busca, isto é, tendendo a um mínimo local. Para que isso não aconteça, cria-se uma rotina a fim de explorar outras áreas do espaço de busca através da mutação aleatória de genes de um cromossomo.

A troca aleatória de posição entre dois genes é feita através do operador Swap.

4.2.6.6 Descrição do Algoritmo

De acordo com Mayerle (1994) e Bezerra (1995) pode-se construir o algoritmo genético com os principais passos:

P1: Construção da população inicial: gere os n cromossomos aleatórios, calcule o custo de todos os cromossomos gerados, construindo uma lista $R = (r_1, r_2, \dots, r_n)$ tal que $C_1 <= C_2 <= \cdots <= C_n$; faça $k = 0$, defina o erro ε e o numero máximo de iterações k_{max};

P2: Teste se $C_n - C_1 \leq \varepsilon$ ou $k \geq k_{max}$, então PARE e apresente r_1;

P3: Seleção natural: elimine o pior cromossomo da lista R, e selecione dois cromossomos $r_p = Select(R)$ e $r_q = Select(R)$ com $r_p \neq r_q$;

P4: Reprodução: faça $r_f = Crossover(r_p, r_q)$;

P5: Mutação: se o cromossomo r_f não é um circuito hamiltoniano, então faça a mutação $r_f = Mutate(r_f)$;

P6: Calcule $F_t = \text{Fitness}(r_f)$ e insira o cromossomo r_f na lista R, mantendo a ordem crescente dos custos; faça $k = k + 1$ e volte ao P2.

ALGORITMO 10 - Algoritmo Genético para o PCV

4.3 GRASP

A metaheurística GRASP (*Greedy Randomized Adaptative Search Procedures*), introduzido por Feo e Resende em 1989, é constituída por heurísticas construtivas e busca local, cada uma iniciando de uma solução diferente. Estas soluções iniciais são geradas por algum tipo de construção randômica gulosa ou algum esquema de perturbação.

A cada iteração, o GRASP, é composto por duas fases: construção da solução e aplicação da busca local na solução construída.

4.3.1 Fase da Construção

Nessa fase uma solução factível é construída elemento a elemento de forma iterativa. Inicialmente o elemento esta em uma lista de candidatos (LC). Através de um fator α (alfa), onde $\alpha \, \epsilon \, [0,1]$, é criada uma lista restrita de candidatos (LRC) que possui os melhores elementos de LC. O tamanho de LRC é determinado por:

$$\text{Cardinalidade (LRC)} = \alpha * \text{Cardinalidade (LC)} \qquad (6)$$

A diversificação da solução depende da quantidade de elementos da lista LRC. Após a definição da LRC, através de algum critério guloso, ou aleatório,

seleciona-se um elemento da mesma para compor a solução. Após a adição do elemento a solução, o processo continua com a atualização de ambas as listas LC e LRC. Quando a cardinalidade de LC for igual à zero, o processo de construção é finalizado.

O algoritmo a seguir apresenta o processo de construção:

Procedimento construção

$S \leftarrow \{\}$

Enquanto solução não completa **faça:**

$LRC = \{c \in C | g(c) \leq s_1 + \alpha(s_2 - s_1)\}$

$c = Seleciona\_Elemento\_Aleatório\ (LRC)$

$S = S \cup \{c\}$

Fim enquanto

Fim Construção

$s_1 = \min\{g(t), t \in C\}$

$s_2 = \max\{g(t), t \in C\}$

$\alpha \in (0,1)$

ALGORITMO 11 - Algoritmo de Construção (GRASP)

O valor de α influencia na qualidade e diversificação da solução gerada na fase da construção (FEO, RESENDE, 1995).

4.3.2 Fase da Melhoria

Esta fase consiste em refinar a solução encontrada na fase anterior aplicando um método de busca local. Corresponde a uma intensificação na solução encontrada explorando regiões vizinhas com o objetivo de encontra um ótimo local. Quanto melhor for a solução encontrada na primeira fase maior será a velocidade para encontrar um ótimo local na fase de busca local. A busca local pode ser sofisticada, porem sem esquecer o diferencial do GRASP que é amostrar o espaço com gerações rápidas. O GRASP é simples e rápido e pode ser integrado a outras técnicas de busca.

4.4 BUSCA TABU

De acordo com Fraga (2006) a metaheurística conhecida como Busca Tabu foi proposta independentemente, por dois autores. Glover em 1986 e Hansen em 1986. O método Busca Tabu é uma metaheurística não estocástica com a capacidade de escapar de ótimos locais. A capacidade de fuga de ótimos locais na Busca Tabu é possível devido a aceitação de movimentos não aprimorantes, junto com a estruturas flexíveis de memória que armazenam informações coletadas durante o processo de busca.

A Busca Tabu utiliza um conceito de memória de curto prazo chamado de Lista Tabu, que guarda informações sobre as ultimas soluções exploradas, e recusa-se a revisitar essas soluções por um determinado tempo. Apenas as principais características das soluções são armazenadas, as soluções completas não são armazenadas, pois o custo computacional é muito alto.

É necessário definir alguns elementos importantes:

- **Espaço de busca**: é o espaço de todas as possíveis soluções que podem ser consideradas durante a busca
- **Vizinhança**: é um elemento fortemente relacionado ao espaço de busca. É o conjunto de todas as soluções encontradas a cada iteração, a partir de modificações que podem ser aplicadas a solução atual.
- **Tabus**: Elementos que previnem visitas cíclicas no espaço de busca. Os tabus são armazenados na lista e possuem um número fixo e limitado de entradas.

O **tamanho da lista** é um parâmetro importante para o método, pois define quantos movimentos farão parte da lista. Se o tamanho for muito grande, o espaço de busca é mais diverso, porém pode restringir demais a busca pois poucos movimentos podem ser executados. Se a lista for muito aumentam a probabilidade de ciclos. Outro parâmetro a ser definido pelo método é o número de iterações em que um movimento estará proibido, chamado de **tenure**. A definição do tamanho da lista e tenure estão relacionadas. Fraga (2006) coloca o tamanho da lista igual ao tenure.

Um problema com o método da Busca Tabu é que a criação de lista de movimentos proibidos pode bloquear movimentos que se executados novamente, gerariam soluções melhores. Portanto uma função de aspiração pode tornar possível a aceitação de um movimento pertencente à lista tabu. Um exemplo de critério de

aspiração é o caso em que a execução de um movimento tabu gere uma solução que tenha o valor da função objetivo melhor do que todas as outras encontradas.

Os critérios de paradas mais comuns são:

- **Número de iterações sem melhora**: após determinado números de iterações sem melhora, o processo termina;
- **Tempo de processamento**: após um tempo de processamento pré-determinado o processo termina.
- **Limite a atingir**: o processo encerra quando um limite pré-definido é alcançado.

O pseudocódigo a seguir representa o algoritmo da Busca Tabu, segundo Brazil, Leite e Mendes (2006):

```
So ← Solução Inicial
S ← So
S* ← So
Lista Tabu ← { }
Enquanto Critério de parada não satisfeito faça:
    Encontrar melhor solução S′ ∈ Vizinhança(S) e ∉ Lista Tabu
    Se custo(S′) < custo(S*) então
        S* ← S′
    Senão
        Se custo(S′) > custo(S*) então
            Lista Tabu ← Lista Tabu ∪ {S}
        Fim Se
    Fim Se
    S ← S′
Fim Enquanto
Fim Busca Tabu
```

ALGORITMO 12 - Algoritmo Busca Tabu

5. INTELIGÊNCIA DE ENXAMES (*SWARM INTELLIGENCE*)

A inteligência de enxames é uma técnica de inteligência computacional que estuda o comportamento coletivo de agentes descentralizados. Estes são indivíduos de uma população que interagem localmente uns com os outros e com o meio ambiente.

O termo “enxame” é utilizado de forma genérica para se referir a qualquer coleção estruturada de agentes capazes de interagir. Esse conceito pode ser facilmente estendido a outros sistemas com arquitetura similar, Como por exemplo uma colônia de formigas, um enxame de abelhas, ou um bando de pássaros em revoada. Podemos ainda considerar uma multidão de pessoas ou um engarrafamento, onde respectivamente os agentes são as pessoas e carros. Um sistema imunológico pode ser considerado um enxame de células e moléculas. Outro exemplo é uma economia, considerada um enxame de agentes econômicos. Na natureza, varias espécies se beneficiam do comportamento social, contribuindo para o aumento da probabilidade de buscar e conseguir alimentos, capacidade de reprodução e o fator de proteção contra os predadores. (ZUBEN; ATTUX, 2008).

No fim da década de 1980, surge o termo “SWARM INTELLIGENCE” referindo-se a sistemas robóticos compostos por uma coleção de agentes simples em um ambiente interagindo de acordo com as regras locais, proposto por Beni e Wang. Pesquisadores desenvolveram ferramentas computacionais para a solução de problemas e seleção de estratégias de coordenação e controle de robôs baseados nas vantagens que certos indivíduos possuem ao viver coletivamente (ZUBEN; ATTUX, 2008).

A Inteligência coletiva ou de enxame é propriedade de sistemas compostos por agentes que possuem pouca ou nenhuma inteligência e com capacidade individual limitada, capazes de apresentar comportamentos coletivos inteligentes, de acordo com White e Pagurek (1998).

Segundo Bonabeau, Dorigo e Theraulaz (1999), qualquer algoritmo ou dispositivo para solução de problemas inspirados no comportamento coletivo de insetos e outras sociedades animais, é considerada como inteligência de enxame.

Algumas características devem ser satisfeitas para que o enxame possua inteligência (MILLONAS, 1994):

Princípio de proximidade: os membros do enxame devem ter noção de tempo e espaço a fim de realizar mudanças sem interferir na movimentação de seus vizinhos.

Princípio de qualidade: Os membros do enxame devem ser capazes de se adaptar aos fatores de qualidade do ambiente, garantindo a qualidade do enxame na busca da melhor solução para determinado objetivo.

Princípio de resposta diversificada: o enxame não deve focar todos os seus recursos em um único canal, as mudanças são desejadas apenas para que o enxame possa evoluir sua busca, evitando que membros da população fiquem presos em regiões onde não é possível encontrar melhores soluções. Mas devem possibilitar medidas corretivas caso o processo evolutivo de busca tenha uma aumento repentino em seu esforço.

Princípio de estabilidade: o enxame deve ser capaz de avaliar a mudança em membros do enxame com a finalidade de melhorar a evolução de sua busca.

Princípio de adaptabilidade: a população deve alterar seu comportamento quando houver necessidade, ou seja, possuir a capacidade de avaliar as mudanças ocorridas.

Normalmente no enxame, não existe um controle centralizado para o comportamento dos indivíduos, embora as interações locais entre eles muitas vezes gerem um comportamento global padronizado. Este padrão comportamental é denominado Emergência – "*Emergence*". De acordo com Steven Johnson (JOHNSON, 2001) este controle pode ser encontrado em vários sistemas de comunicação, cidades e sistemas de seres vivos. Johnson, em seu livro Emergence, define como emergência o que acontece quando várias entidades independentes de baixo nível conseguem, sem ter estratégia ou autoridade centralizada, criar uma organização de alto nível. Esse padrão de comportamento é chamado de "*botton up*" onde cada indivíduo ou entidade acaba inconscientemente determinando o comportamento coletivo do grupo, sem haver liderança.

Para a sobrevivência de um enxame, existem tarefas que precisam ser realizadas em sincronia e de foram contínua, como por exemplo, um enxame de abelhas que possuem uma vida social extremamente complexa com tarefas pré-definidas, como coleta, produção de alimento, proteção, entre outras. Essas tarefas são vitalmente importantes para a sobrevivência do grupo. Entretanto mesmo sem um comando centralizado, a colmeia se mantém organizada e funcional.

5.1 OTIMIZAÇÂO POR COLÔNIA DE FORMIGAS

Segundo Glover e Kochenberger (GLOVER, KOCHENBERGER, 2003), *Ant Colony Optimization* (ACO) é uma abordagem inspirada no uso que as formigas fazem dos feromônios como um canal de comunicação, isto é, é um sinal químico liberado por um animal que dispara uma resposta natural em outros membros da mesma espécie.

Basicamente uma formiga sai aleatoriamente em busca por alimento e marca seu trajeto através de feromônios. Cada colônia possui um cheiro único que distingue as formigas de um ninho e de outro. Os feromônios informam alguns comportamentos específicos como qual o tipo de trabalho a ser feito, se há perigo, se existe a necessidade de recrutar operárias para carregar alimento até a colônia entre outros.

Ao encontrar alimento ela reforça esse caminho, assim outras formigas ao encontrarem esse trajeto tem maior probabilidade de segui-lo, assim estabelecendo um caminho “eficiente” até a comida, e reforçam ainda mais esse caminho.

5.1.1 Heurística *Ant System*

A heurística conhecida por Ant System (AS) foi desenvolvida por Colorni e colaboradores (COLORNI *et al.*, 1991, 1992), Baseadas no estudo do comportamento de uma colônia de formigas. No processo de levar comida até o formigueiro. A ideia é utilizar um conjunto de agentes que se comunicam entre si para resolver problemas de otimização (DORIGO *et al.*, 1996).

Considerando os seguintes aspectos:

- As formigas são capazes de encontrar o caminho do formigueiro até o local do alimento e voltar ou contornar obstáculos com relativa facilidade.
- Estudos descobriram que esta capacidade é o resultado de uma espécie de comunicação química entre estes insetos, utilizando uma substancia chamada feromônios.

O fenômeno é causado pela presença simultânea de muitas formigas.

5.1.2 PCV Baseado no Algoritmo de AS

Considere o PCV definido em um grafo $G(N, A)$, com $|N|$ nós e $|A|$ arcos, onde os nós n_i e $n_j$são conectados pelo arco (i, j) e este é associado a um valor real $d_{ij} = d(n_i, n_j)$. O problema consiste em encontrar uma permutação π dos nós que minimizam: $\sum_{i=1}^{|N|} d(n_{\pi(i)}, n_{\pi(i+1)})$ onde $\pi(|N|+1) \equiv \pi(1)$, isto é, procura-se uma sequência de vértices que forneça o menor circuito Hamiltoniano possível.

A idéia é modelar a construção de uma trilha de feromônios produzidas pelas formigas. Define-se então que a formigas são agentes com as seguintes propriedades:

- As formigas lembram as cidades já visitadas através de uma lista ordenada, denominada lista tabu (LT). Observar que não trata-se do algoritmo de Busca Tabu.
- A cada passa as forminhas escolhem uma cidade para se mover, dentre aquelas que não estão na lista tabu, de acordo com uma regra probabilística;
- Após a construção de uma determinada rota, a formiga atribui à substância feromônio τ_{ij}, de cada arco (i, j) usado.

No caso do PCV, a regra probabilística usada como função de seleção é o procedimento de Monte Carlo, onde a probabilidade de escolher a cidade j como destino é dada por:

$$\text{Prob(J)} = \frac{\tau_{ij}^{\alpha} . \eta_{ij}^{\beta}}{\sum_{l \in LT} \tau_{il}^{\alpha} . \eta_{il}^{\beta}} \qquad (7)$$

Onde:

- τ_{ij} é a intensidade de feromônio no arco (i, j);
- $\eta_{ij} = \frac{1}{d_{ij}}$ é chamado de visibilidade do nó j a partir do nó i;
- $\rho (0 \leq \rho \leq 1)$ é um parâmetro do algoritmo tal que $(1 - \rho)$ pode ser interpretado como fator de evaporação do feromônio;
- τ é um parâmetro do algoritmo tal que inicialmente as formigas estão livres para se mover aleatoriamente.
- α e β são parâmetros que permitem o controle do impacto relativo dos critérios de visibilidade e intensidade de feromônio.

Sendo Q uma constante positiva, os passos do algoritmo são:

P1: Faça $\eta_{ij} = \frac{1}{\mathrm{d_{ij}}}; \tau_{ij} = \tau$

P2: Para cada formiga k(k=1 até M), faça

$\mathrm{LT_k} = (\text{começar com uma cidade para a formiga k})$

Enquanto $|\mathrm{LT_k}| < |\mathrm{N}|$ faça

Selecione uma cidade j para se deslocar

Adicione j em ordem na lista $\mathrm{LT_k}$

$\mathrm{L_k} = \text{distância total do percurso } \mathrm{LT_k}$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{\mathrm{Q}}{\mathrm{L_k}} & \text{se o arco (i, j)pertence ao percurso descrito em } \mathrm{LT_k} \\ 0 & \text{caso contrário} \end{cases}$$

$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$

P3: $\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij}$

P4: se não foi atendida a condição de termino, retorne ao passo 2.

ALGORITMO 13 - Algoritmo Ant System para o PCV

Dentre as diversas versões do *Ant System*, três se destacam:

- A primeira consiste em um procedimento de construção que começa com um número pequeno de cidades e uma a uma as outras cidades vão sendo introduzidas. Permitindo assim quer as formigas identifiquem boas distribuições de feromônio para cada subproblema, que serão utilizados como base para novos subproblemas;
- A segunda consiste em modificar o nível de feromônio quando uma estagnação é detectada. A modificação é obtida, com a troca em um pequeno número de iterações, dos parâmetros α e β.
- A terceira consiste em designar valores específicos para α e β para cada formiga, fazendo-os evoluir por algoritmos genéticos, com uma função de *fitness* inversamente proporcional ao comprimento total da rota. Onde os parâmetros podem ser codificados por cadeias binárias e os operadores genéticos clássicos podem ser utilizados.

5.2 OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS (PSO)

A metaheurística de otimização por enxame de partículas, PSO, *Particle Swarm Optimization* do inglês, foi introduzida por pelo psicólogo James Kennedy e pelo engenheiro eletricista Russel Eberhart (KENNEDY; EBERHART, 1995) na década de 90. Esta fundamentada na observação do comportamento social de algumas espécies de pássaros e foi modelada pelo biólogo Frank Heppner (HEPPNER; GRENANDER, 1990).

Por ser uma técnica inspirada na natureza pode ser considerada como uma técnica de computação evolutiva, pois é baseada em mecanismos biológicos com o objetivo de solucionar computacionalmente problemas de otimização combinatória. Em um algoritmo evolutivo, uma das características é a existência de uma população de soluções, que contém a solução para o problema abordado.

No algoritmo do PSO cada indivíduo é denominado partícula, e utiliza a sua própria experiência e a experiência do enxame para atingir o objetivo, baseado em informações globais e locais para tomar suas decisões. Algumas propriedades do comportamento social como imitação, avaliação e comparação, são base para o enxame de partículas e são utilizados para a resolução de problemas e na adaptação a mudanças no meio ambiente. (KENNEDY, EBERHART, 2001).

Sendo considerado um sistema multiagente, é importante ressaltar a forma como os indivíduos se comunicam. Eles apresentam um comportamento colaborativo e cooperativo entre si, compartilhando experiências na busca de uma melhor solução.

No PSO, as partículas percorrem o espaço de busca através da combinação entre a melhor posição encontrada por ela mesma e a melhor posição encontrada pelas partículas vizinhas do enxame a cada iteração. A vizinhança é o conjunto com as quais a partícula em questão pode se comunicar, podendo este conjunto ser todo o enxame. O PSO executa um conjunto de operadores, movimentando cada partícula no espaço de busca. Através da aplicação da velocidade, que é ajustada a cada iteração com base na melhor posição encontrada pela vizinhança da partícula e pela melhor posição da própria partícula, as partículas são alteradas criando-se assim um novo conjunto de soluções, como se as partículas fossem movimentadas em um espaço n-dimensional.

5.2.1 Terminologia no Algoritmo do PSO

Os principais termos utilizados no PSO são descritos a seguir (REYES-SIERRA, COELHO, 2006):

- **Enxame ou Swarm**: população do algoritmo;
- **Partícula** (X_{id}): Indivíduo da população, que representa uma possível solução e possui uma determinada posição;
- **Vetor Velocidade** (V_{id}): o vetor velocidade determina a direção que a partícula se movimentará, visando melhorar a sua posição atual;
- $\boldsymbol{P_{best}}$ (P_{id}): Melhor posição já encontrada pela partícula;
- $\boldsymbol{L_{best}}$ (P_{ld}): Melhor posição já encontrada por uma partícula pertencente à vizinhança de determinada partícula;
- $\boldsymbol{G_{best}}$ (P_{gd}): Melhor posição já encontrada por uma partícula em toda a população;
- **Líderes**: partículas da população que possuem os melhores valores na função objetivo;
- **Coeficiente de Inércia** (w): responsável pela influência de valores anteriores no cálculo da velocidade atual;
- **Fator de individualidade** (c_1): responsável por influenciar a partícula com relação à melhor posição encontrada por ela mesma;
- **Fator de sociabilidade** (c_2): responsável por influenciar a atração da partícula em direção à melhor posição encontrada por qualquer partícula de sua vizinhança;
- **Fator de constrição**: (χ) similar ao coeficiente de inércia, também é responsável pela influência de valores anteriores no cálculo da velocidade atual.
- **Topologia de Vizinhança**: responsável pela estrutura de comunicação de uma partícula com sua vizinhança.
- **Fitness**: Determina o desempenho de uma partícula através do calculo da função objetivo.

5.2.2 Estruturas do Algoritmo PSO

No algoritmo original, o sistema é inicializado com uma população randômica, que se move em um espaço de busca n-dimensional sendo cada uma delas uma possível solução. Além disso, cada partícula possui uma velocidade randômica. A i-ésima partícula é presentada por um vetor n-dimensional dada por:

$$X_{id} = (X_{i1}, X_{i2}, \dots, X_{in}) \tag{8}$$

A melhor posição encontrada pela i-ésima partícula é representada por:

$$P_{id} = (P_{i1}, P_{i2}, \dots, P_{in}) \tag{9}$$

A melhor posição encontrada por uma partícula pertencente a vizinhança da i-ésima partícula é representada por:

$$P_{ld} = (P_{l1}, P_{l2}, \dots, P_{ln}) \tag{10}$$

O vetor velocidade ou a taxa de variação de posição para cada partícula é representada por:

$$V_{id} = (V_{i1}, V_{i2}, \dots, V_{in}) \tag{11}$$

O movimento de cada partícula é baseado em três fatores: sociabilidade, individualidade e velocidade. As equações (12) e (13) determinam as movimentações das partículas:

$$V_{id}^{t} = V_{id}^{t-1} + c_1 \cdot \varepsilon_1 \cdot (P_{id} - X_{id}) + c_2 \cdot \varepsilon_2 \cdot (P_{ld} - X_{id}) \tag{12}$$

$$X_{id}^{t} = X_{id}^{t-1} + V_{id} \tag{13}$$

Onde: c_1 e c_2 são constantes positivas correspondentes aos fatores de individualidade e sociabilidade respectivamente, e ε_1 e ε_2 correspondem a valores aleatórios com distribuição uniforme no intervalo [0,1].

Cada partícula sofre a influência de três vetores que podem ser descritos como:

- Vetor Inércia: representado na equação (12) por V_{id}, a velocidade que direciona a partícula para a região onde acredita-se que estão melhores soluções;
- Vetor memória: representado na equação (12) pelo termo $c_1 \cdot \varepsilon_1 \cdot (P_{id} - X_{id})$. É a componente cognitiva que relaciona a posição atual e a melhor posição encontrada pela própria partícula;
- Vetor Cooperação: representado na equação (12) pelo termo $c_2 \cdot \varepsilon_2 \cdot (P_{ld} - X_{id})$. É a relação entre a posição atual da partícula e a melhor posição encontrada pela sua vizinhança.

A equação (12) é responsável por atualizar a velocidade da partícula de acordo com sua velocidade anterior, sua variação de posição atual com sua melhor posição e com a melhor posição do seu vizinho. Já equação (13) atualiza a posição da partícula.

No PSO clássico, para a inicialização do processo, deve ser definida uma população inicial com uma determinada quantidade de partículas do enxame. O PSO consiste em uma mudança de velocidade e posição das partículas a fim de que elas encontrem suas posições de P_{best} e L_{best} a cada iteração, através da avaliação do seu desempenho (*fitness*). Além disso, deve existir uma condição de parada para o processo, podendo ser um determinado número de execuções ou ainda uma estagnação na melhora do desempenho. O processo do PSO pode ser verificado na Figura 5.

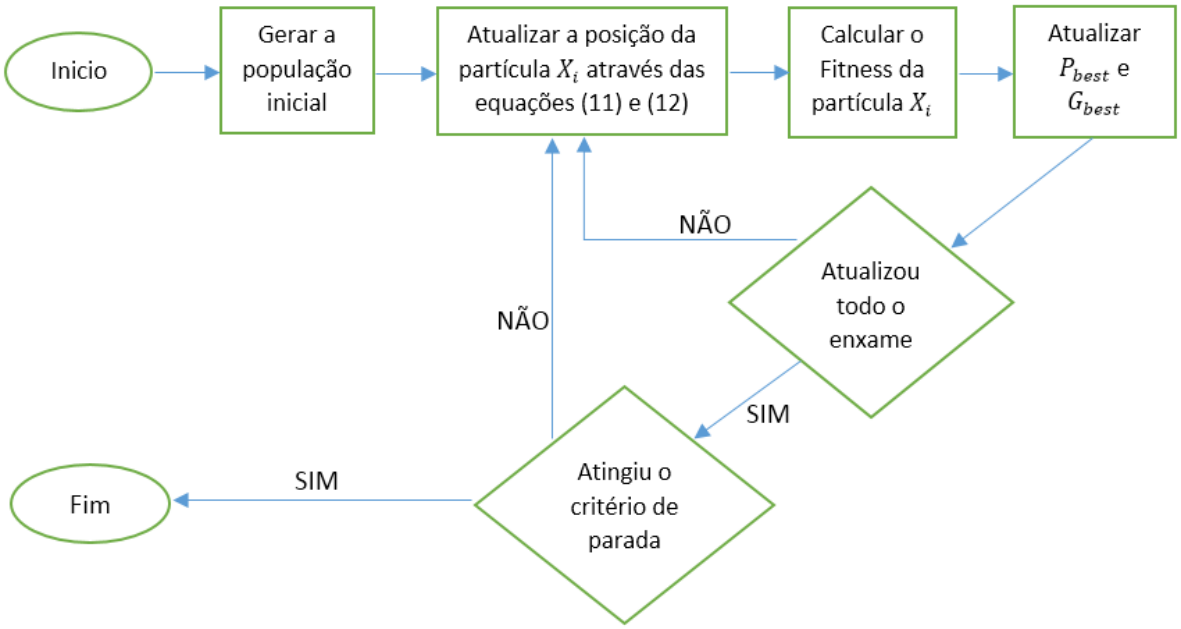

FIGURA 5: FLUXOGRAMA REPRESENTANDO O ALGORITMO DE PSO

FONTE: O AUTOR (2014)

O esquema gráfico da Figura 6 representa as influências desses vetores no processo de atualização da partícula.

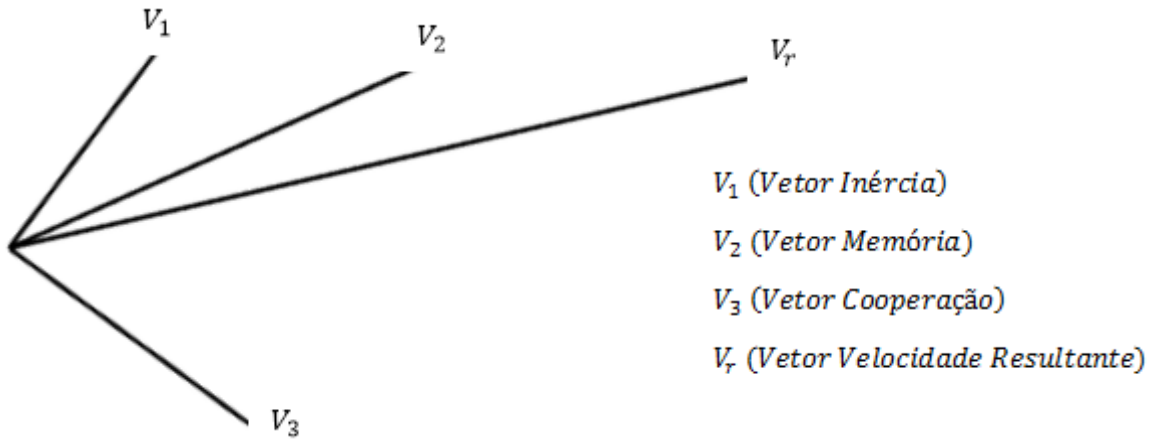

FIGURA 6: VETORES QUE INLUENCIAM A ATUALIZAÇÃO DA PARTÍCULA

FONTE: O AUTOR (2014)

5.2.3 Peso de Inércia e Fator de Constrição

Um fenômeno que acontece no algoritmo clássico, é que facilmente a partícula adquire uma velocidade alta e acaba oscilando dentro do espaço de busca.

Eberhart e Kennedy (1995) propuseram um mecanismo para estabelecer um limite máximo para a velocidade da partícula. Porém a escolha incorreta do valor da velocidade pode implicar na queda de desempenho, ou seja, espaços de busca maiores necessitam de uma velocidade máxima maior e espaços de busca menores precisam de uma velocidade de busca menor.

Na intenção de remover o conceito de velocidade limite, Clerk e Kennedy (2002) introduziram o conceito de peso de inercia e fator de constrição, reescrevendo a equação (12) de atualização da velocidade para a equação (14).

$$V_{id}^{t} = w \cdot V_{id}^{t-1} + c_1 \cdot \varepsilon_1 \cdot (P_{id} - X_{id}) + c_2 \cdot \varepsilon_2 \cdot (P_{ld} - X_{id}) \qquad (14)$$

Na execução do algoritmo, inicialmente o valor inicial de w, deve ser 1, para favorecer o comportamento de exploração inicial da partícula, e em seguida deve ser reduzido, com a finalidade de realizar uma busca mais refinada, já que supostamente a partícula já movimenta-se por regiões mais próximas a solução, de acordo com os autores.

O fator de constrição é a introdução de um novo parâmetro χ, similar ao peso de inércia, derivado das constantes existentes na equação (12) de atualização da velocidade.

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \qquad \varphi = c_1 + c_2 \qquad (15)$$

Aplicando o fator de constrição, obtemos a equação (16), de atualização da velocidade:

$$V_{id}^{t} = \chi(w \cdot V_{id}^{t-1} + c_1 \cdot \varepsilon_1 \cdot (P_{id} - X_{id}) + c_2 \cdot \varepsilon_2 \cdot (P_{ld} - X_{id})) \qquad (16)$$

Foi demonstrado pelos autores que para valores de $c_1 + c_2 = \varphi < 4$, o enxame converge lenta e espiralmente, entretanto, quando $\varphi > 4$ a convergência é rápida e garantida.

5.2.4 *Heterogeneous* PSO

No algoritmo clássico do PSO, a nuvem de partículas é homogênea, isto é, todas as partículas têm a mesma quantidade de vizinhos e as regras para atualização de velocidade e posição das partículas, os modelos de influência e os parâmetros são os mesmos para todas as partículas (DE OCA, 2009).

Dependendo da natureza das diferenças entre as configurações das partículas, diferentes tipos de heterogeneidade podem ser identificados. Se as partículas mudam as configurações de acordo com o tempo são consideradas *dynamics* (dinâmicas), caso contrário, são consideradas *static* (estáticas). Caso algum comportamento da nuvem de partículas provoque alteração de uma configuração, são chamados enxames adaptáveis (*adaptive)*. Existem quatro tipos de heterogeneidade devido aos diferentes aspectos de configuração das partículas:

A heterogeneidade de vizinhança: aparece quando as partículas têm quantidades diferentes de vizinhos. Esse tipo de heterogeneidade ocorre quando a topologia de vizinhança não é um grafo regular, e permite que algumas partículas potencializem mais a influência no processo de busca coletiva do que outras. O tamanho da nuvem de partículas tem grande efeito no nível de vizinhança heterogênea.

Modelo de influência heterogênea: aparece quando as partículas possuem diferentes mecanismos na escolha de seus informantes. Por exemplo, se algumas partículas forem informadas apenas pelas melhores partículas de sua vizinhança enquanto outras podem ser completamente informadas.

Regras de Atualização heterogêneas: se diferentes partículas possuem diferentes regras para atualização de posição no espaço de busca.

Parâmetros heterogêneos: devem ser satisfeitas duas condições para que os parâmetros (i) um grupo de partículas deve ter as mesmas regras de atualização; (ii) e pelo menos um par dessas partículas devem diferenciar os parâmetros de atualização.

5.2.5 Topologias.

A topologia do enxame de partículas defini como a troca de informações entre os indivíduos é efetuada. Esta troca de informações tem influência significativa no desempenho do algoritmo. Por isso algoritmos de otimização possuem mecanismos próprios, responsáveis pela troca de informação entre seus indivíduos, buscando sempre melhorar o desempenho desses. A velocidade das partículas também é influenciada pela forma de comunicação entre as mesmas. As variações das estruturas, topologias, de comunicação são responsáveis pelo desempenho do algoritmo. Dependendo do problema a ser tratado, na busca de uma solução ótima, pode-se priorizar a velocidade de convergência e/ou a qualidade da solução. Existem vários tipos de topologias, as topologias global e local são as mais comumente usadas em algoritmos do PSO para problemas discretos. Para problemas não discretos existem outras topologias, como por exemplo, topologia focal, hierárquica, Von Neumann, multi-ring, four-cluster, clan e *random*.

5.2.5.1 Topologia Global

Foi a primeira topologia proposta, e de acordo com a sua estrutura, todas as partículas do enxame estão conectadas. Dessa forma cada partícula recebe informação do L_{best} de todo o enxame. Esta topologia permite uma convergência mais acelerada. Essa estrutura apresenta um desempenho adequado para problemas que admitem apenas um mínimo global. Porém existe o risco da partícula convergir para um mínimo local, não explorando desta forma todo o espaço de busca, portando em problemas multimodais, apesar de sua convergência ser rápida, não se pode garantir a qualidade da solução encontrada. O grafo da Figura 7 ilustra esse tipo de topologia.

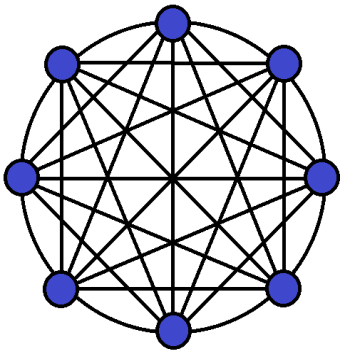

FIGURA 7: TOPOLOGIA GLOBAL
FONTE: O AUTOR (2014)

5.2.5.2.Topologia Local

A topologia local é a mais importante variação da topologia global, onde cada partícula se comunica apenas com seus vizinhos diretos. Normalmente apresenta um melhor desempenho em problemas multimodais por explorarem melhor o espaço de busca não se prendendo a mínimos locais.

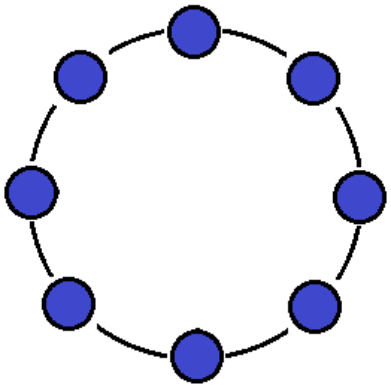

FIGURA 8: TOPOLOGIA LOCAL
FONTE: O AUTOR (2014)

Porém em problemas unimodais, sua convergência é inferior se comparada à topologia global, pois num enxame de n partículas, cada partícula pode esperar ate $\frac{n}{2}$ iterações para receber informações da melhor partícula. Sua estrutura é apresentada na Figura 8.

5.2.5.3 Topologia Focal

Na topologia focal, uma determinada partícula apresenta um comportamento específico no enxame, estando conectada a todas as outras partículas, porém uma partícula não focal está conectada apenas à partícula focal. Esta partícula focal só atualiza sua posição caso haja uma melhora no seu desempenho e é responsável, a cada iteração, pela análise e decisão se as informações das partículas não focais conectadas a elas devem repassadas para o enxame.

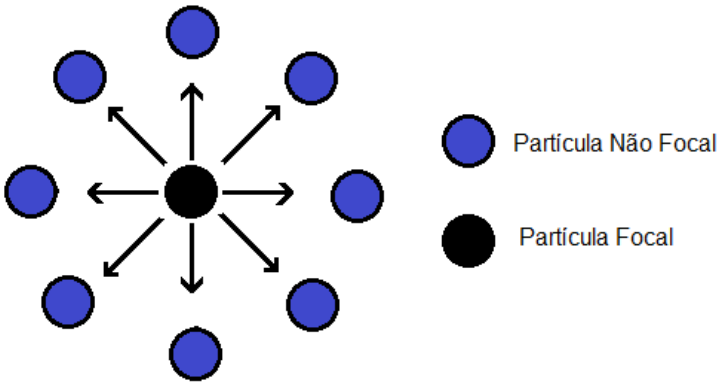

FIGURA 9: TOPOLOGIA FOCAL
FONTE: O AUTOR (2014)

Esta topologia produz um movimento sobre influência da melhor partícula do enxame com poucas execuções operacionais. Considerando um enxame com n partículas e caso a partícula focal repasse a informação a todas as partículas do enxame, esta topologia comporta-se como uma topologia global com $n - 1$ partículas. O grafo da Figura 9 representa essa topologia.

5.2.5.4 Topologia Hierárquica

De acordo com essa topologia as partículas estão organizadas em forma de arvore, como mostra o grafo da Figura 10.

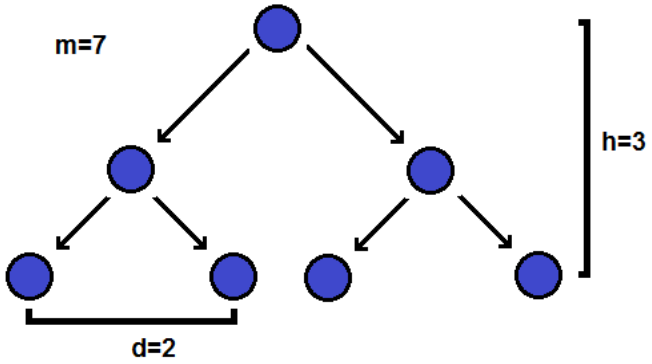

FIGURA 10: TOPOLOGIAL HIERÁRQUICA
FONTE: O AUTOR (2014)

Cada nó deste grafo possui apenas uma partícula, que é influenciada por sua melhor posição até o momento, P_{best}, e pela posição da partícula que esta logo acima neste grafo, chamada de partícula pai. O algoritmo calcula o desempenho (fitness) entre a partícula pai e seus filhos, caso a partícula de nível inferior, partícula filho, encontre uma posição melhor que a partícula pai, é feita a troca de posição entre essas partículas. Esse processo é executado antes da atualização da velocidade e posição. Essa estrutura topológica tem como vantagem o rápido acesso às melhores soluções, porém sua estrutura torna a busca complexa.

Nessa estrutura, considerando h como a altura da árvore hierárquica, d a sua ordem e m o número de nós da estrutura, cada partícula pode subir apenas um nível hierárquico por iteração e pode chegar ao topo em h-1 iterações.

5.2.5.5 Topologia Von Neumann

A topologia Von Neumann (KENNEDY; MENDES, 2002) apresenta um grafo fechado comparado a uma malha, com vizinhança estática de quatro vizinhos diretos (esquerdo, direito, superior e inferior), de acordo com a figura 11, onde as partículas de uma extremidade estão conectadas a partículas de extremidades opostas.

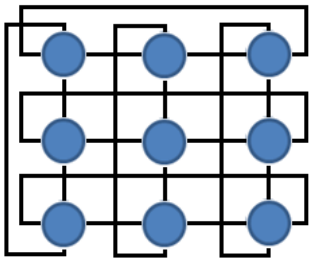

FIGURA 11: TOPOLOGIA VON NEUMANN

FONTE: O AUTOR (2014)

5.2.5.6 Topologia Multi-Ring

A topologia multi-ring (BASTOS, 2008) é baseada na topologia local, consistindo na combinação de múltiplos anéis de partículas, onde cada anel é composto pelo mesmo número de partículas, e cada partícula do anel pode se comunicar com vizinhos diretos, ou seja, partículas de anéis adjacentes. Considerando um conjunto de n+1 anéis, onde o k-ésimo anel é denominado $rl_{(k)}$, e cada partícula i pertencente a este anel, representada por $rl_{(k)(i)}$, troca informações com as partículas $rl_{(k+1)(i)}$e $rl_{(k-1)(i)}$. Porém os anéis $rl_{(o)}$ e $rl_{(n)}$ tem uma comunicação limitada, pois sua vizinhança não é completa.

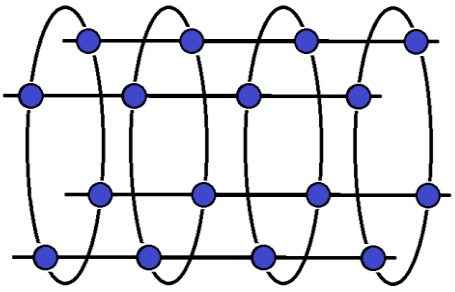

FIGURA 12: TOPOLOGIA MULT-IRING

FONTE: O AUTOR (2014)

Na topologia Multi-Ring os anéis possuem a habilidade rotacional, se o desempenho do algoritmo não for melhorado, com isso a partícula passa a ter uma nova vizinhança. Porém se os anéis não rotacionarem essa topologia se resumirá à topologia Von Neumann. Essa topologia melhora a capacidade de exploração do espaço de busca e ao mesmo tempo fornece uma melhor qualidade na solução final, e é apresentada na figura 12.

5.2.5.7 Topologia Four-Clusters

De acordo com Mendes, Kennedy e Neves (2002) essa topologia é inspirada em pequenos agrupamentos, chamados de clusters de partículas, que se comunicam através de partículas informantes ou líderes. A quantidade de clusters está diretamente ligada à quantidade de partículas líderes em cada cluster, considerando um grupo de n clusters, devem existir $n-1$ informantes em cada cluster. Cada partícula informante permanece a mesma durante toda a execução do algoritmo, ou seja, a informação se propaga de informante para informante. A informação das partículas não é informada diretamente aos outros clusters (A, B, C, D), como pode ser verificado na Figura 13. Porém isso ocasiona iterações sem garantia de melhorias.

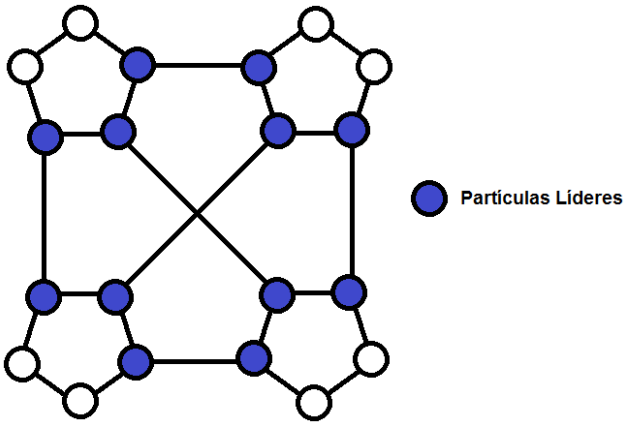

FIGURA 13: TOPOLOGIA FOUR-CLUSTERS
FONTE: O AUTOR (2014)

5.2.5.8 Topologia Clan

Representada na figura 14, a topologia Clan, é semelhante à topologia Four-Clusters. Porém existe apenas uma partícula líder, determinada internamente, e que tende a guiar as outras partículas. De acordo com Kennedy e Eberhart (1995) as partículas de cada clã se comunicam através da topologia global. A cada iteração, o clã marca a partícula que obteve o melhor resultado na solução. Os líderes escolhidos formam um novo enxame, e o processo de busca é executado considerando apenas a informação dos líderes. Os líderes podem se comunicar tanto de maneira global quanto local.

Após a atualização da velocidade e posição dos líderes, esses repassam as informações às outras partículas de seus respectivos clãs, portanto todas as partículas serão influenciadas indiretamente pela melhor partícula do enxame.

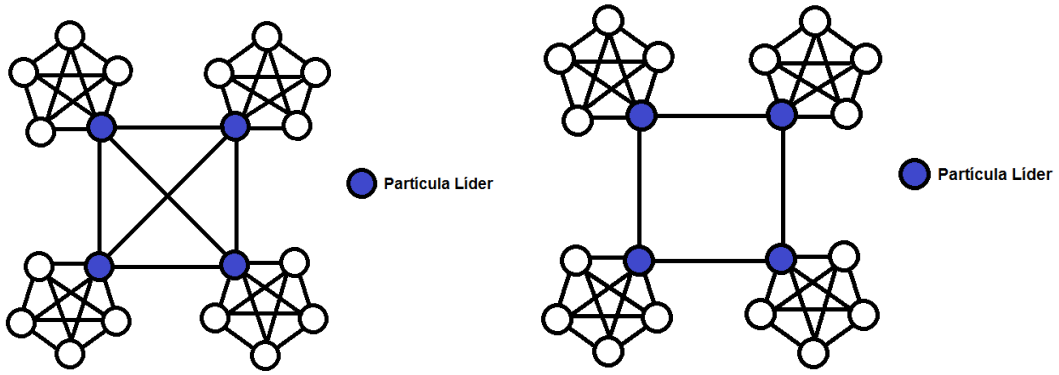

FIGURA 14: (A)TOPOLOGIA CLAN GLOBAL, (B) TOPLOLOGIA CLAN LOCAL
FONTE: O AUTOR (2014)

5.2.5.9 Topologia *Random*

De acordo com Clerc (2007), essa topologia tem um mecanismo que determina uma vizinhança aleatória para uma determinada partícula. A partícula tem influência em si mesma.

Considere um enxame com n partículas, definido por $S = \{P_1, P_2, \dots, P_n\}$, cada partícula P_i possui, incluindo a si mesma, K informantes. Por exemplo, na topologia global $K = n$ e na topologia local $K = 3$. Para a escolha da vizinhança V, de P_i algumas regras devem ser satisfeitas:

$P_i \subset V$, isto é, a vizinhança contém P_i;

As outras $K - 1$ partículas P_j, da vizinhança são escolhidas aleatoriamente de S.

Com essa escolha aleatória existe a possibilidade da partícula ter influência apenas de si mesma, possibilitando uma busca mais profunda na região de busca onde ela se encontra.

O comportamento desta topologia segue o algoritmo a seguir:

Construir uma matriz $N \times N$ denominada M, onde a diagonal principal é inicializada com 1. Significando que cada partícula influenciará a si mesma.

Para cada linha i de M, sorteia-se aleatoriamente N números $n_1, n_2, \dots, n_n$ e inicializa-se $M(i, n_n) = 1$;

Considerando cada coluna j, se $M(i, j) = 1$, a partícula P_i informa a partícula P_j.

Se após determinado número de iterações a melhor partícula não melhora seu desempenho, sua vizinhança é alterada. Além disso, pode-se estabelecer como critério para a mudança de vizinhança um determinado número de iterações. Como esta estratégia tem um custo muito alto, o reorganização do enxame ocorre após $\frac{N}{2}$ iterações.

6. PSO PARA PROBLEMAS DISCRETOS

O algoritmo clássico foi inicialmente proposto para resolução de problemas contínuos, tendo sido modificado, de modo a resolver problemas discretos. As modificações propostas para o PSO discreto preservam toda a estrutura do PSO original.

A primeira versão do algoritmo do PSO discreto foi proposta por Kennedy e Eberhart (1997). Os autores codificaram as partículas, $X_{id} = (x_{i1}, x_{i2}, \dots, x_{in})$, como uma sequência binária e a velocidade, $V_{id} = (v_{i1}, v_{i2}, \dots, v_{in})$, como um vetor de probabilidades.

6.1 OPERADORES DISCRETOS

Para que o PSO pudesse ser aplicado ao PCV, Clerc (2004) definiu alguns elementos do PSO:

- Posição

O problema PCV considera o grafo $G = (N, A)$, onde N é o conjunto dos vértices (nós) desse grafo, representando as cidades, numerados de 1 até N, e A é o conjunto de triplas $(i, j, c_{i,j})$, para i, j variando de 1 ate N e $c_{i,j}$, que representa os arcos de ligação entre as cidades .

- Partícula

Considera-se uma partícula como a sequência com $N + 1$ nós, $X = (x_1, x_2, \dots x_N, x_{N+1})$ com $x_1 = x_{N+1}$, e $x_i \neq x_j \ \forall i, j \in \{2, \dots, N\}$ Esta partícula só será viável se existirem todos os arcos $(x_i, x_{i+1}), \forall i \in \{1, \dots, N + 1\}$.

- Função Fitness

Para cada partícula pode-se definir a função objetivo como sendo a soma dos pesos dos arcos, ou especificamente, a soma das distâncias entre as cidades adjacentes determinadas pela rota encontrada por cada partícula.

$$f(x) = \sum_{i=1}^{N} c_{i,j}\,, \qquad j \in \{1, \dots, N\}$$

Como a busca é pelo caminho de custo mínimo, esse problema é de minimização.

- Velocidade

A velocidade é definida por uma lista de pares de vértices, que indicam a sequência de transposições a ser aplicada na partícula X. A lista transposições de pares é definida como uma permutação de N elementos, onde $\|V\|$ é o comprimento desta lista.

$$V = \big((i_k, j_k)\big) \ \therefore i_k, j_k \in \{1, \dots, N\}, k \in \{1, \dots, \|V\|\}$$

Ou seja, troque os números (i_1, j_1), depois troque (i_2, j_2) e assim sucessivamente até $\big(i_{\|V\|}, j_{\|V\|}\big)$.

- Movimentação (Posição e Velocidade)

Considerando uma partícula X e uma velocidade V, a partícula $X^t = X^{t-1} + V$ é definida aplicando a primeira transposição de V em X, a segunda ao resultado e assim até completar toda a lista de transposições.

Exemplo: $X^{t-1} = (1,2,3,4,5,6,1)$ e $V = ((1,2),(3,4))$, $\|V\| = 2$

Portanto: aplicando $(1,2)$ em X obtemos $X = (2,1,3,4,5,6,2)$, em seguida aplicamos $(3,4)$ em X, obtemos $X^t = (2,1,4,3,5,6,2)$

- Obtenção da Velocidade

Supondo duas partículas X_1 e X_2, a velocidade é definida como a diferença $X_2 - X_1$, isto é, a lista de transposições necessárias para construir a partícula X_2 a partir da partícula X_1. O algoritmo usado para encontrar a velocidade deve satisfazer:

$X_2 - X_1 = \neg(X_1 - X_2)$ e $X_1 = X_2 \Rightarrow X_2 - X_1 = \emptyset$

Onde $\neg$ denota que $X_2 - X_1$ e $X_1 - X_2$ geram listas com as mesmas transposições porem na sequência de ordem inversa. E $\emptyset$ representa uma lista vazia de transposições

- Adição entre velocidades

Considerando duas velocidades V_1 e V_2. Define-se $V_1 + V_2$, como a união das listas de transposições de V_1 e V_2, respectivamente nesta ordem. Particularmente esta operação é definida tal que $V_1 + \neg V_1 = \emptyset$.

Ex.: $V_1 = ((1,2),(3,4))$ e $V_2 = ((2,3),(41))$,

Portanto $V_1 + V_2 = ((1,2),(3,4),(2,3),(4,1))$

- Multiplicação entre coeficiente e velocidade.

Considerando $c \in \Re$, e V uma velocidade.

- Se $c = 0$, $c \cdot V = \emptyset$
- Se $c \in]0,1] \Rightarrow$ trunca-se v, isto é, considera-se $\|c \cdot V\|$ como o maior inteiro menor ou igual a $c \cdot \|V\|$, então define-se $c \cdot V = \big((i_k, j_k)\big), k \in \{1, \dots, \|V\|\}$.
- Se em que $c > 1$, significa ter $c = k + c', K \in \aleph^*, c' \in]0,1]$, então se define $c \cdot V = \underbrace{V + \dots + V}_{k\ vezes} + c' \cdot V$.

- Distância entre duas partículas.

Sejam as partículas X, X e X_3. A distância entre X_1 e X_2 é definida por $d(X_1, X_2) = \|X_2 - X_1\|$, satisfazendo as seguintes propriedades:

a) $\|X_2 - X_1\| = \|X_1 - X_2\|$

b) $\|X_2 - X_1\| = 0 \Rightarrow X_1 = X_2 = 0$

c) $\|X_2 - X_1\| \leq \|X_2 - X_3\| + \|X_3 - X_1\|$

Alguns autores,(WANG *et al,* 2003), apresentaram um algoritmo com a mesma estrutura básica de Clerc, porém a partícula possui a representação com apenas N nós e não mais $N + 1$ nós em sua sequência. A função objetivo para N cidades é:

$$f(x) = \sum_{i=1}^{N} w_{n_i, n_{i+1}} + w_{N,1}$$

A velocidade também foi definida de maneira diferente. A troca é feita entre as posições e não mais pelos números definidos no vetor velocidade.

Portanto para o algoritmo de Wang temos o seguinte cálculo:

Exemplo: $X = (1,2,3,4,5,6)$ e $V = ((1,2),(2,4))$, $\|V\| = 2$

Portanto: aplicando $(1,2)$ em X obtemos $\bar{X} = (2,1,3,4,5,6)$, em seguida aplicamos $(2,4)$ em $\bar{X}$, obtemos $X' = (2,4,3,1,5,6)$.

O algoritmo Set-Based PSO, desenvolvido por Chen e*t al* (2009) desenvolveu um algoritmo de PSO baseado na teoria de conjuntos e probabilidades. A principal característica é a forma como foi implementada a velocidade, definida como um

conjunto de possibilidades. Se E é uma sequência binária representando uma solução do problema, e V um conjunto de possibilidades definido em E, dado por:

$$V = \left\{ \frac{e}{p(e)} \middle| e \in E \right\},$$

onde tal que cada elemento $e \in E$ tem uma possibilidade $p(e) \in [0,1]$ em V.

Chen *et al* (2009) usou a teoria de conjuntos para definir a operação de obtenção de velocidade, porém a atualização de velocidade foi feita através da equação (14) do PSO. Além de outras regras para definir a multiplicação entre coeficiente e partícula a as posições das partículas.

O algoritmo clássico do PSO discreto não apresentou resultados satisfatórios para o Problema do Caixeiro-Viajante, foram então desenvolvidas e implementadas outras estratégias para melhorar os resultados obtidos, como a busca local e o Path-relinking.

6.2 ALGORITMO PSO - PATH-RELINKING

O *Path-relinking* é uma técnica de intensificação e diversificação, proposta por Glover e Laguna (1997). O algoritmo consiste em gerar um caminho entre uma solução inicial x_i e uma solução objetivo x_o, criando novas soluções intermediárias. A solução x_o deve pertencer à vizinhança de x_i, e é chamada de solução guia (*guide solution*). Para gerar o caminho é necessário aplicar movimentos progressivos em x_i com a finalidade de diminuir a distância entre x_i e x_o. O caminho é uma sequência de soluções $x_i(1), x_i(2), \dots, x_i(r) = x_o$ onde $x_i(l+1)$ é criada a partir de $x_i(l)$ a cada passo, por meio de movimentos que reduzam o número de movimentos para atingir x_o (GLOVER;LAGUNA, 2000).

Um problema frequentemente encontrado em algoritmos do PSO é que o mesmo pode estagnar em ótimos locais. Portanto, foi introduzido um fator de dispersão no algoritmo, que recria a nuvem de partículas após um determinado número de iterações $(nuvemDispersa)$, mantendo as k melhores partículas.

No PSOPR, a velocidade é um operador que cobre os três movimentos em uma única operação (ROSENDO; POZO, 2010).

$$V_{id} = \overbrace{w \cdot \underbrace{V_{id}}_{M1}}^{Busca\ Local} + \overbrace{c_1 \cdot \varepsilon_1 \cdot \underbrace{(P_{id} - X_{id})}_{M2} + c_2 \cdot \varepsilon_2 \cdot \underbrace{(P_{ld} - X_{id})}_{M3}}^{Path-relinking}$$

onde P_{id} é a melhor posição encontrada pela partícula X_{id} e P_{ld} é a melhor posição encontrada por uma partícula da vizinhança da partícula X_{id}

Dessa forma, no PSOPR, a velocidade é definida como um conjunto de três operadores aplicados a cada partícula, assim como no PSO clássico.

Rosendo e Pozo (2010) sugerem que a ordem de execução dos movimentos seja $M1$, $M3$ e $M2$. Primeiro a busca local $(M1)$ é realizada até o limite definido por w, então o *path-relinking* para o movimento $(M3)$ é aplicado e em seguida o *path-relinking* é aplicado ao movimento $(M2)$. Nos movimentos $M2$ e $M3$, os limites são estabelecidos por $c_1 \cdot \varepsilon_1$ e $c_2 \cdot \varepsilon_2$, respectivamente.

O Path-relinking foi utilizado na implementação dos movimentos $M2$ e $M3$, baseado na proposta inicial de Glover e Laguna (1997). A solução inicial X_i e a solução objetivo foram chamadas de P_{ini} como sendo a posição atual da partícula e P_{obj} a posição da partícula objetivo. A P_{obj} pode representar P_{best} e L_{best} se os movimentos forem $M2$ e $M3$ respectivamente.

Para a execução de movimento da P_{ini} em direção a P_{obj} é necessário o cálculo da distância entre estas soluções. Rosendo e Pozo (2010) sugerem a utilização da função distância exata $(distExata)$, definida por Ronald (1998) por ser um método computacionalmente barato, descrito como o total de dimensões diferentes entre si. Por exemplo:

$$P_{ini} = \{1, \mathbf{2}, 5, \mathbf{4}, \mathbf{3}, 6\}$$

$$P_{obj} = \{1, \mathbf{3}, 5, \mathbf{2}, \mathbf{4}, 6\}$$

Como $P_{ini}^2 = 2$ e $P_{obj}^2 = 3$, $P_{ini}^4 = 4$ e $P_{obj}^4 = 2$, $P_{ini}^5 = 3$ e $P_{obj}^5 = 4$, temos diferença nas dimensões 2, 4 e 5, ou seja, 3 dimensões diferentes entre si, portanto:

$$distExata(P_{ini}, P_{obj}) = 3$$

Rosendo e Pozo (2010) ainda usam o coeficiente de individualidade (c_1) igual ao coeficiente de sociabilidade (c_2), assim como os coeficientes aleatórios ε_1 e ε_2. Portanto:

$$c = c_1 = c_2 \text{ e } \varepsilon = \varepsilon_1 = \varepsilon_2$$

É necessário também calcular a quantidade de movimentos a serem aplicados, a qual chama-se $calculaPassos$, dada pela seguinte equação:

$$calculaPassos = distExata(P_{ini}, P_{obj}) * f(c, \varepsilon)$$

onde,

$$f(c, \varepsilon) = \frac{c + \varepsilon}{2}$$

Rosendo e Pozo ainda introduziram o parâmetro $c_{pr} = (0, 1]$ que limita o número de soluções intermediárias avaliadas durante o *Path relinking*, pois se todas as soluções forem avaliadas o custo computacional poderá ser muito alto.

O pseudocódigo do algoritmo 14, a seguir, apresenta como o *Path relinking* foi utilizado neste trabalho. Primeiramente são calculados a distância e o número de passos que serão executados e o intervalo entre as avaliações. Em seguida, a cada iteração, uma alteração é aplicada a P_{ini}. O algoritmo verifica se o intervalo de passos sem avaliar a solução é atingido. E em caso positivo P_{ini} é avaliada. Se for verificado que esta nova posição é melhor que P_{best}, então a posição atual de P_{ini} é guardada como P_{best}. Apos isso, o contador de avaliações é incrementado. Após todas as iterações, é retornada como nova posição da partícula, a posição alcançada na última troca.

$distância \leftarrow distExata(P_{obj}, P_{ini})$

$totalPassos \leftarrow calculaPassos(distância, c, \varepsilon)$

$Avaliações = 0;\ Passo = 1;\ intervalo = \frac{1}{c_{pr}}$

Para $passo = 1$ até $totalPassos$ **faça**:

escolha a dimensão d

aplique a alteração em P_{ini}^{d}

se $passo \geq intervalo * avaliações$ **então**:

avalie P_{ini}

se P_{ini} é melhor que P_{obj} **então**:

$P_{best} = P_{ini}$

Fim se

Incremente avaliações

Fim se

Fim Para

Retorne P_{ini}

ALGORITMO 14 - PSOPR (*Path-relinking*)

O pseudocódigo do algoritmo PSOPR-com dispersão (algoritmo 15) é apresentado a seguir.

Atribui parâmetros

Enquanto $i = 0$ até $tamanhoEnxame$ **faça:**

Inicia X_i com valores aleatórios

$P_{best} \leftarrow L_{best} \leftarrow X_i$

Fim enquanto

Enquanto condição de parada não satisfeita **faça:**

Enquanto $dis = 0$ menor $nuvemDispersa$ **faça**

Para $i = 0$ até $tamanhoEnxame$ **faça**

$X_i \leftarrow buscalocal(X_i)(M1)$

$X_i \leftarrow pathRelinking(X_i, P_{ld})(M3)$

$X_i \leftarrow pathRelinking(X_i, P_{id})(M2)$

$P_{ld} \leftarrow$melhor entre X^ie P_{ld}

$P_{id} \leftarrow$melhor entre X^ie P_{id}

Fim Para

$Reinicia\ nuvem$

Fim enquanto

Fim enquanto

Retorna melhor P_{ld}

ALGORITMO 15 - PSOPR-Dispersão

7. IMPLEMENTAÇÃO E EXPERIMENTOS COMPUTACIONAIS

Este capítulo descreve a implementação do algoritmo PSOPR para o PCV. Assim como a configuração final do algoritmo.

Os experimentos foram executados em um computador Intel Core I5 (1.4 GHz e 8 Gb de RAM) com sistema operacional Ubuntu 13.10. e implementados em C++

7.1 IMPLEMENTAÇÃO E PARAMETROS PARA O PCV

O Algoritmo do PSOPR implementado utiliza o algoritmo LK para a busca local, descrito anteriormente. Neste trabalho foi usada a versão implementada no *software Concorde TSP Solver*[1].

Devido ao fato do PCV ser um problema modelado em forma de grafo, é impossível modificar apenas uma dimensão da partícula e manter sua viabilidade, dessa forma para executar uma alteração na partícula é utilizado o operador (*swap*).

O operador *swap* realiza trocas simples entre duas dimensões aleatórias, isto é inicialmente escolhe uma dimensão a tal que $P_{obj}^{a} \neq P_{ini}^{a}$, após escolhe uma dimensão b tal que $P_{obj}^{b} = P_{ini}^{b}$ então troca os valores contidos em P_{ini}^{a} e P_{ini}^{b}.

No PCV a função distância exata $(distExata)$ é uma estimativa do número de posições intermediárias que podem ser geradas entre P_{ini} e P_{obj}. Isto não causa problemas para gerar soluções, pois a própria função que calcula o número de passos leva em consideração o coeficiente ε, que tem como função variar a quantidade de trocas aplicadas à partícula.

O coeficiente de inércia (w) foi configurado de forma dinâmica, decrescendo linearmente. Foi utilizada a definição proposta por Shi e Eberhart (1998), onde:

$$w = \frac{(iMax - i) * \left(w_{final} - w_{inicial}\right)}{iMax} + w_{inicial}, \quad i = 1,2,3, \dots, iMax \qquad (17)$$

Onde $iMax$ representa o número máximo de iterações e i é a iteração atual.

[1] http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm

Dessa forma, inicialmente, as partículas tendem a seguir mais o seu caminho, e a sofrer menos influência de outras partículas (L_{best}) e de sua própria experiência (P_{best}).

A variação dos fatores de individualidade c_1 e sociabilidade c_2, de acordo vários pesquisadores, devem ser idênticos (EBERHART; SHI, 2000).

As simulações foram realizadas utilizando o algoritmo PSOPR, com o objetivo de avaliar o desempenho das topologias Global, Local, Focal, Von Neumann e Clan descritas anteriormente na subseção 5.2.5

7.2 INSTÂNCIAS DO PCV E PARÂMETROS DE SIMULAÇÃO

O algoritmo foi aplicado a 8 instâncias simétricas do PCV, descritas no TSPLIB[2], variando de 76 a 1002 cidades. Os nomes dessas instâncias são: EIL76, LIN105, PR144, RAT195, PR299, PR439, D657 e PR1002

Para essas instâncias do PVC foram realizadas 30 execuções em cada topologia. De acordo com Chen *et al* (2009) os algoritmos de PSO aplicados ao PCV tendem a gerar bons resultados com um número de partículas variando entre 20 e 40. Após algumas simulações foi definido o tamanho da nuvem igual a 30 partículas. O número máximo de iterações foi configurado em 1000. Foram selecionadas as 10 melhores partículas para reiniciar a nuvem após a dispersão.

Os coeficientes de individualidade e de sociabilidade foram configurados $c_1 = c_2 = 0{,}8$. O coeficiente que determina quantas soluções intermediárias serão avaliadas no Path-relinking $c_{pr} = 0{,}2$ e a inércia (w) foi configurada para variar de $0{,}9$ até $0{,}4$.

Para os experimentos sem a estratégia de dispersão, foram considerados como critério de parada: atingir o número máximo de 1000 iterações; ou atingir no máximo 100 iterações sem melhora na solução, ou atingir o valor ótimo. Para os experimentos com a estratégia de dispersão, o objetivo é analisar essa estratégia, com os dois primeiros critérios mantidos.

[2] https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

7.3 EXPERIMENTOS COM O PSOPR – PCV

Os resultados desses experimentos são apresentados nas tabelas 1 e 4.

As tabelas mostram o resultado para o conjunto de instâncias simétricas. São apresentadas nas colunas da tabela o nome da instância, a melhor solução alcançada (Min) e a solução média (Méd), o desvio padrão (DP) em termos de desvio percentual, e o tempo computacional (T). A média representa a media aritméticas das soluções mínimas encontradas nas 30 execuções, o Min é a diferença percentual entre a melhor solução encontrada pelo algoritmo e a melhor solução conhecida e o tempo de execução em segundos para cada topologia analisada. O desvio percentual é calculado de acordo com a equação (18), onde Sol representa a melhor solução ou a média das soluções encontradas e Opt representa a solução ótima.

$$(Sol - Opt) \times \frac{100}{Opt} \tag{18}$$

O valor 0,0000 indica que a solução ótima foi alcançada.

TABELA 1: COMPARAÇÃO ENTRE AS TOPOLOGIAS GLOBAL E LOCAL SEM DISPERSÃO

Topologia	Global				Local			
Instância	Min(%)	Méd(%)	DP(%)	T(s)	Min(%)	Méd(%)	DP(%)	T(s)
ELI76	0,0000	0,0000	0,0000	8,3881	0,0000	0,0000	0,0000	9,8747
LIN105	0,0000	0,0000	0,0000	10,440	0,0000	0,0000	0,0000	10,2091
PR144	0,0000	0,0000	0,0000	22,0767	0,0000	0,0000	0,0000	29,3034
RAT195	0,0000	0,0000	0,0000	33,8131	0,0000	0,0000	0,0000	42,0711
PR299	0,0000	0,0000	0,0000	61,2136	0,0000	0,0000	0,0000	58,6424
PR439	0,0000	0,0467	0,0856	354,453	0,0000	0,0424	0,0337	462,8720
D657	0,3015	0,3235	0,0612	648,1330	0,3192	0,4338	0,0443	713,2556
PR1002	0,9145	0,9146	0,0545	1367,7029	0,9964	1,2059	0,0860	976,5184
MÉDIA	**0,1520**	0,1606	0,0251	313,2777	0,1644	0,2102	0,0205	287,8433

TABELA 2: COMPARAÇÃO ENTRE AS TOPOLOGIAS GLOBAL E LOCAL COM DISPERSÃO

Topologia	Global				Local			
Instância	Min(%)	Méd(%)	DP(%)	T(s)	Min(%)	Méd(%)	DP(%)	T(s)
EIL76	0,0000	0,0000	0,0000	7,6120	0,0000	0,0000	0,0000	8,6660
LIN105	0,0000	0,0000	0,0000	9,1025	0,0000	0,0000	0,0000	9,1739
PR144	0,0000	0,0000	0,0000	25,6873	0,0000	0,0000	0,0000	25,4508
RAT195	0,0000	0,0000	0,0000	38,2812	0,0000	0,0000	0,0000	37,8110
PR299	0,0000	0,0000	0,0000	54,4777	0,0000	0,0000	0,0000	52,5041
PR439	0,0000	0,0273	0,0252	405,4701	0,0000	0,0341	0,0291	420,7356
D657	0,2394	0,2764	0,0446	754,7840	0,2759	0,3474	0,0363	644,5944
PR1002	0,7652	0,8149	0,0462	1195,2127	0,8335	0,9871	0,0744	1145,0111
MÉDIA	**0,1256**	0,1398	**0,0145**	311,3284	0,1387	**0,1711**	0,0175	292,9934

TABELA 3: COMPARAÇÃO ENTRE AS TOPOLOGIAS FOCAL, VON NEUMANN E CLAN SEM DISPERSÃO												
Topologia	Focal				Von Neumann				Clan			
Instância	Min(%)	Med(%)	DP(%)	T(s)	Min(%)	Med(%)	DP(%)	T(s)	Min(%)	Med(%)	DP(%)	T(s)
EIL76	0,0000	0,0000	0,0000	9,0393	0,0000	0,0000	0,0000	6,1689	0,0000	0,0000	0,0000	6,7663
LIN105	0,0000	0,0000	0,0000	8,2241	0,0000	0,0000	0,0000	8,6024	0,0000	0,0000	0,0000	13,6155
PR144	0,0000	0,0000	0,0000	22,859	0,0000	0,0000	0,0000	27,8582	0,0000	0,0000	0,0000	29,9994
RAT195	0,0000	0,0000	0,0000	43,752	0,0000	0,0000	0,0000	44,8901	0,0000	0,0000	0,0000	42,1601
PR299	0,0000	0,0000	0,0000	44,915	0,0000	0,0000	0,0000	45,8072	0,0000	0,0000	0,0000	60,6530
PR439	0,0000	0,0450	0,0822	351,9753	0,0000	0,0838	0,0654	363,1899	0,0000	0,1059	0,1129	424,149
D657	0,2985	0,3106	0,0652	7541562	0,2883	0,3246	0,0927	638,6737	0,3591	0,5125	0,0777	546,5438
PR1002	1,0647	1,1122	0,0508	1250,4251	1,0884	1,1669	0,0588	1186,0180	1,0873	1,1110	0,0601	827,3141
MÉDIA	0,1704	0,1969	0,0247	310,6684	0,1702	0,1834	0,0271	290,1510	0,1808	0,2162	0,0312	243,9002

TABELA 4: COMPARAÇÃO ENTRE AS TOPOLOGIAS FOCAL, VON NEUMANN E CLAN COM DISPERSÃO												
Topologia	Focal				Von Neumann				Clan			
Instância	Min(%)	Med(%)	DP(%)	T(s)	Min(%)	Med(%)	DP(%)	T(s)	Min(%)	Med(%)	DP(%)	T(s)
EIL76	0,0000	0,0000	0,0000	7,8035	0,0000	0,0000	0,0000	7,1303	0,0000	0,0000	0,0000	7,8278
LIN105	0,0000	0,0000	0,0000	9,1436	0,0000	0,0000	0,0000	9,9192	0,0000	0,0000	0,0000	11,8997
PR144	0,0000	0,0000	0,0000	26,1510	0,0000	0,0000	0,0000	25,0057	0,0000	0,0000	0,0000	27,1387
RAT195	0,0000	0,0000	0,0000	39,6075	0,0000	0,0000	0,0000	38,8988	0,0000	0,0000	0,0000	38,3093
PR299	0,0000	0,0000	0,0000	51,0481	0,0000	0,0000	0,0000	53,7004	0,0000	0,0000	0,0000	54,9582
PR439	0,0000	0,0380	0,0708	406,4471	0,0000	0,0684	0,0524	418,8807	0,0000	0,0893	0,0945	366,4184
D657	0,2422	0,2614	0,0528	653,2939	0,2478	0,2701	0,0746	565,4048	0,2908	0,4296	0,0622	491,4368
PR1002	0,9185	0,9610	0,0437	1085,9556	0,9076	0,9839	0,0499	1074,9837	0,8697	0,9302	0,0515	725,7061
MÉDIA	0,1451	0,1576	0,0209	284,9313	0,1444	0,1653	0,0221	274,2409	0,1451	0,1811	0,0260	215,4619

TABELA 5: COMPARAÇÂO DA ESTRATÉGIA COM DISPERSÃO X SEM DISPERSÃO

Topologia	Global	
	Com Dispersão	Sem Dispersão
Min (%)	0,1256	0,1520
Med (%)	0,1398	0,1606
Tempo (s)	311,3284	313,2777
Topologia	Local	
	Com Dispersão	Sem Dispersão
Min (%)	0,1387	0,1644
Méd (%)	0,1711	0,2102
Tempo (s)	292,9934	287,8433
Topologia	Focal	
	Com Dispersão	Sem Dispersão
Min (%)	0,1451	0,1704
Méd (%)	0,1576	0,1834
Tempo (s)	284,9313	310,6684
Topologia	Von Neumann	
	Com Dispersão	Sem Dispersão
Min (%)	0,1444	0,1720
Méd (%)	0,1653	0,1969
Tempo (s)	274,2409	290,1510
Topologia	Clan	
	Com Dispersão	Sem Dispersão
Min (%)	0,1451	0,1808
Méd (%)	0,1812	0,2162
Tempo (s)	215,4619	243,9002

Nos resultados apresentados nas tabelas 1 e 3, que comparam as topologias sem a estratégia de dispersão, pode-se verificar que para os problemas com menor número de cidades, isto é, até 500 cidades, todas as topologias atingiram o ótimo. Entretanto para os problemas de instâncias com número intermediário de cidades, maior do que 500 cidades, a topologia Global se mostrou 15,8% melhor em relação

aos mínimos e 25,6% em relação aos resultados médios. Porém o tempo de execução foi 44% maior.

Analisando os dados das tabelas 2 e 4 e comparando-se a média para os resultados apresentados, tem-se que a média dos mínimos com a topologia global é 14,1% melhor que todas as outras topologias, assim como a média dos desvios padrão foi 49,1% melhor. Para as soluções médias, a topologia local se mostrou em média 20,6% melhor se comparada às demais topologias. Contudo o tempo médio na topologia Clan, foi em média 34,9% melhor que as outras topologias.

Analisando a tabela 8.5, que compara a média entre o PSO-LK com e sem dispersão, pode-se observar um desempenho 16,7% melhor (em média) para o algoritmo com o critério de dispersão para as instâncias simétricas.

Tem-se que as topologias global e local apresentaram um desempenho 8,1% melhor em relação ao mínimo e a média se comparadas às demais topologias com a estratégia de dispersão e sem a estratégia de dispersão 7,7% melhor. Porém se comparadas em relação ao tempo às topologias focal, Von Neumann e Clan obtiveram 15,4% e 6,3% melhor desempenho para as estratégias com e sem dispersão, respectivamente.

Os grafos das Figuras 15 e 16 representam a solução para a o problema D657 com a técnica sem dispersão e com dispersão, respectivamente, para a topologia de melhor desempenho.

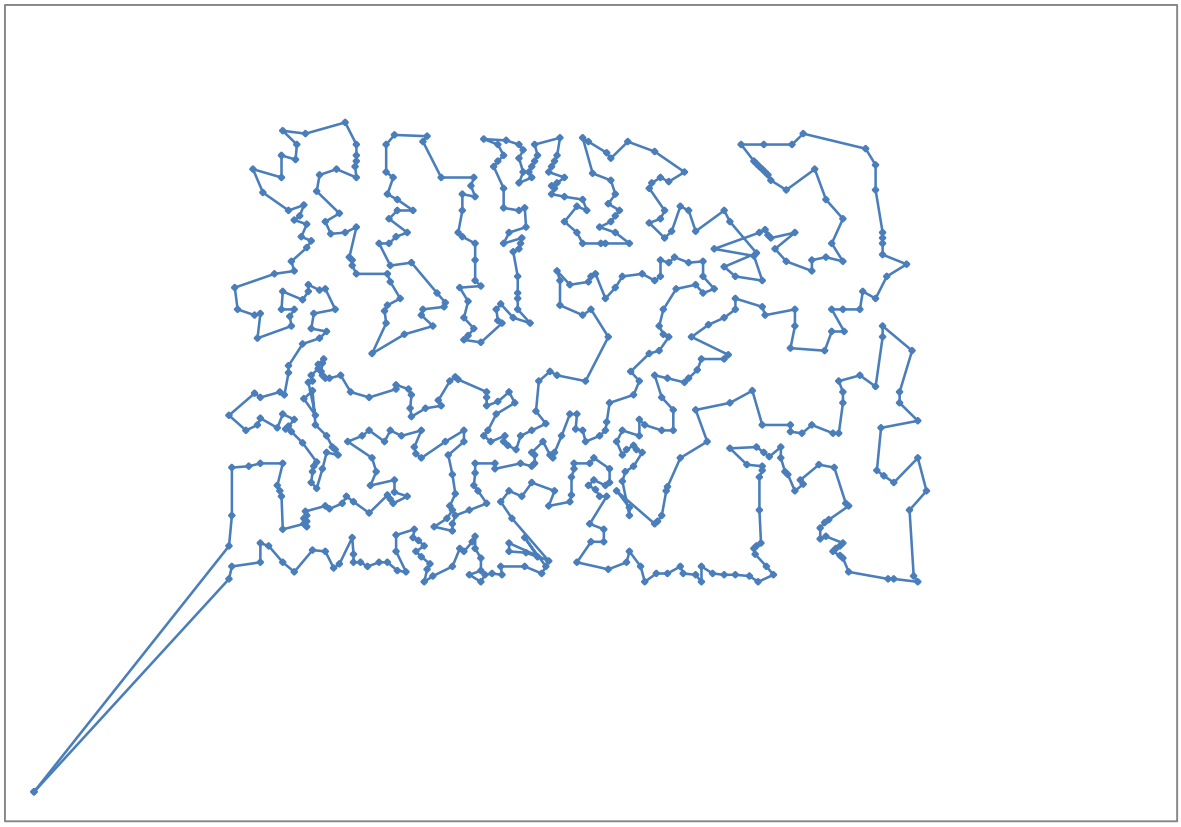

FIGURA 15 - SOLUÇÃO INSTÂNCIA D657 SEM DISPERSÃO

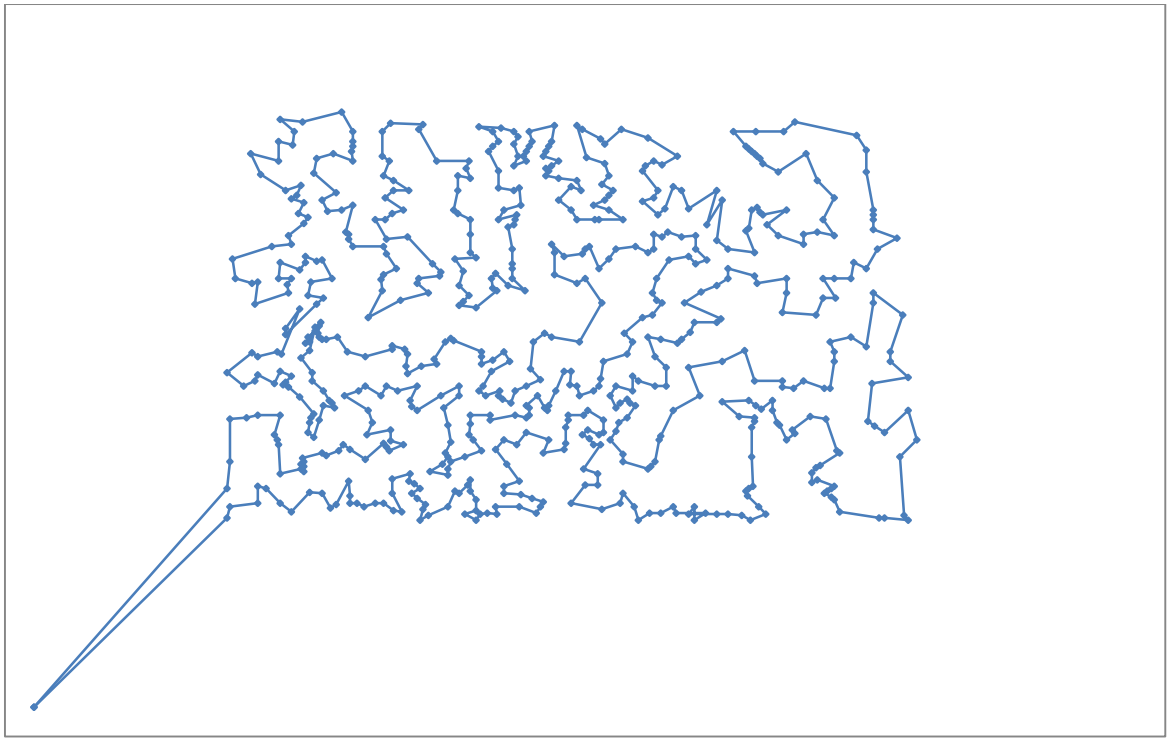

FIGURA 16 - SOLUÇÃO INSTÂNCIA D657 COM DISPERSÃO

Os grafos das Figuras 17 e 18 representam a solução para a o problema PR1002 com a técnica sem dispersão e com dispersão, respectivamente.

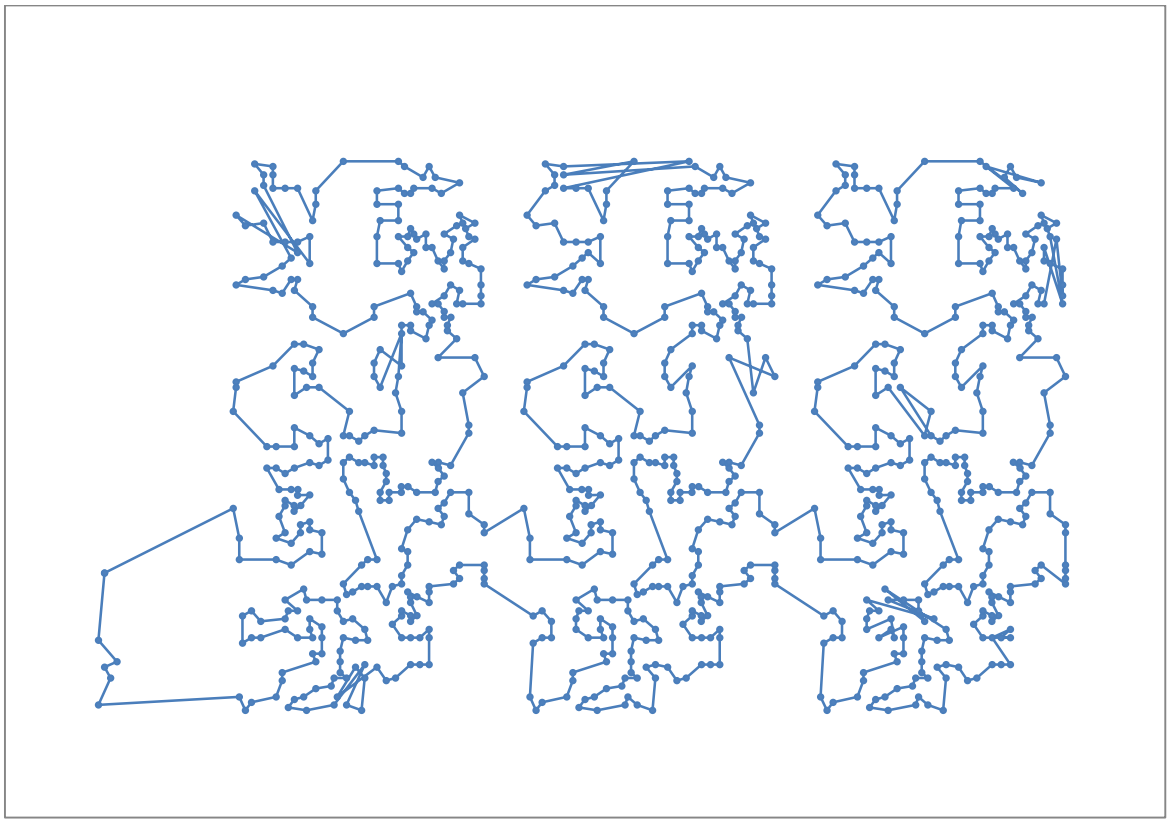

FIGURA 17 - SOLUÇÃO INSTÂNCIA PR1002 SEM DISPERSÃO

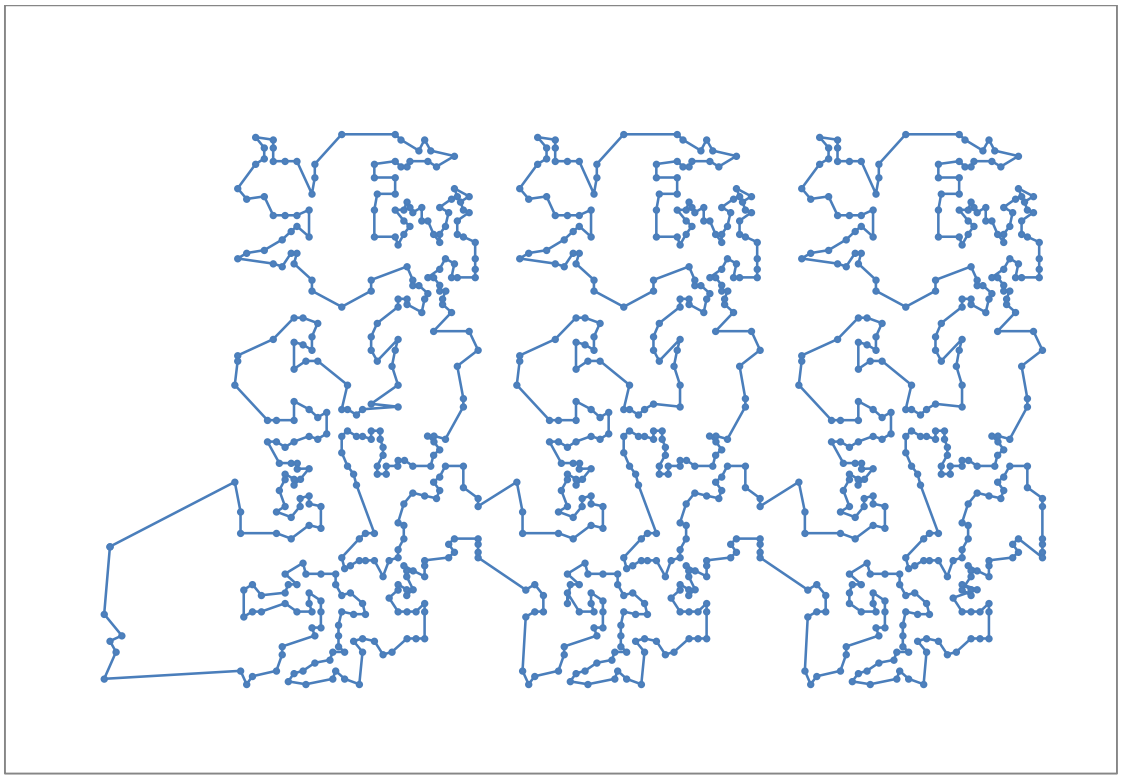

FIGURA 18 - SOLUÇÃO INSTÂNCIA PR1002 COM DISPERSÃO

8. CONSIDERAÇÕES FINAIS

8.1 CONCLUSÃO

A implementação de técnicas matemáticas, para a solução de problemas de otimização, geralmente têm custo computacional elevado. Os pesquisadores Keneky e Heberhart (1995) propuseram o algoritmo de otimização por enxame de partículas com o intuito de resolver de maneira eficiente alguns desses problemas. E dentre os problemas de otimização combinatória, o mais conhecido é o Problema do Caixeiro-Viajante, descrito no capítulo 2.

Quase em sua totalidade, os trabalhos desenvolvidos neste campo dependem de simulações sobre uma determinada proposta, considerando desde uma nova topologia até a análise do efeito da qualidade dos parâmetros adotados.

Os capítulos 3, e 4 apresentaram, de maneira geral, o conceito de otimização, as heurísticas e metaheurística utilizadas na solução do PCV. O capítulos 5 apresenta a inteligência de enxames e a otimização por enxame de partículas, os conceitos necessários para entender seu desenvolvimento, sua estrutura, tanto na versão clássica (contínua). O Objetivo principal deste trabalho consistiu na adaptação das topologias focal, Von Neumann e clan, também apresentadas no capítulo 5, para o algoritmo do PSO discreto. Porém, além destas foram também apresentadas as topologias já conhecidas global, local, hierárquica, multi-ring, four-clusters e *random.*

Este trabalho apresentou nos capítulo 6 e 7 o modelo de PSO discreto a ser implementado, conhecido como PSO-PR assim como a estratégia de dispersão, para a resolução do problema do caixeiro-viajante, que utiliza a busca local para movimentar a partícula o sentido de seguir seu próprio caminho, e o path-relinking para seguir a experiência das demais partículas de sua vizinhança.

Para a validação e experimentação do algoritmo proposto, assim como os detalhes e parâmetros específicos para a implementação do algoritmo e as instâncias simétricas do *TSPLIB* abordadas também foram descritas no capítulo 7. A comparação dos resultados entre o desempenho das variações topológicas assim como da estratégia de dispersão também são neste mesmo capítulo.

A análise dos resultados entre as variações topológicas mostrou que para os problemas simétricos as topologias adaptadas são equivalentes, pois apresentaram

praticamente o mesmo desempenho. As topologias global e local, juntas, apresentam em média um desempenho de aproximadamente 8% melhor do que as outras topologias, com critério de dispersão e aproximadamente 9% melhor sem o critério de dispersão. Porém a um tempo computacional 14,5% maior para a estratégia com dispersão e 6,3% maior sem a estratégia de dispersão. Portanto foi possível verificar que outras topologias são alternativas viáveis, podendo reduzir o custo computacional, assim como a estratégia de dispersão da nuvem se mostrou eficiente.

8.2 TRABALHOS FUTUROS

Como trabalhos futuro, podem ser citados os seguintes tópicos:

1. Adequar o operador de path-relinking, utilizado no PSO-PR, para a utilização de instâncias assimétricas da *TSPLIB*.
2. Adequação das topologias multi-ring e *random* para o PSO discreto.
3. A realização de estudos na solução de outros problemas de otimização combinatória através de outas topologias.
4. A realização de estudos em problemas combinatórios em que a solução ótima não é previamente conhecida.

REFERÊNCIAS

BARBOSA, V.C.; Redes Neurais e Simulated Annealing como Ferramentas para Otimização Combinatória; Investigación Operativa, Vol.1, N.2, 1989.

BARBOSA, H.J.C.; Introdução aos Algoritmos Genéticos; Mini Curso – XX CNMAC, Gramado(RS), 1997.

BARBOZA, A.O.; Simulação e Técnicas da Computação Evolucionária Aplicadas a Problemas de Programação Linear Inteira Mista. Tese de Doutorado, UTFPR, 2005.

BASTOS-FILHO, C. J. A.; CARACIOLO, M.; MIRANDA, P. e CARVALHO, D. Multi Ring Particle Swarm Optimization. Em: Simpósio Brasileiro de Redes Neurais, 2008.

BEZERRA, O.B.; Localização de postos de coleta para apoio ao escoamento de produtos extrativistas - Um estudo de caso aplicado ao babaçu. Dissertação de Mestrado, UFSC, 1995

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, NJ, 1999.

BODIN, L.; GOLDEN, B.; ASSAD, A.; BALL, M.; Routing and Scheduling of Vehicles and Crews – The State of the Art; Computers Ops Res; Vol. 10, Number 2; Pergamon Press, 1983.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR), ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, 2003. ISSN 0360-0300.

BRAZIL, J.C.; LEITE, M.S.; MENDES, R.B.S.; Uma Metaheurística Grasp Busca Tabu para o Problema do Caixeiro Viajante. II Workshop de Computação Científica da UENF - IIWCC, 2006

CARVALHO, A. B. de. Otimização por nuvem de partículas multiobjetivo no aprendizado indutivo de regras: extensões e aplicações. Dissertação (Mestrado) - Universidade Federal do Paraná, 2008.

CHEN, W.-N. et al. A novel set-based particle swarm optimization method for discrete optimization problems. Evolutionary Computation, IEEE Transactions on, v. 14, n. 2, p. 278–300, april 2009. ISSN 1089-778X.

CLERC, M.; e KENNEDY, J. The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space. Em: IEEE Transactions on Evolutionary Computation, v. 6, n. 1, p. 58-73, 2002.

CLERC, M. Discrete particle swarm optimization, illustrated by the traveling salesman problem. In: New Optimization Techniques in Engineering, Springer. Heidelberg, Germany: Springer Berlin / Heidelberg, 2004. P. 219–239.

CLERC, M. Back to Random Topology. Relatório Técnico, mar. 2007, disponível em: < http://clerc.maurice.free.fr/pso/random_topology.pdf > acesso em 20 de novembro de 2012.

COLORNI, A.; DORIGO, M.; MANIEZZO, V.; Distributed Optmization by Ant Colonies; Proceedings of ECAL-91-European Conference on Artificial Life, Paris, France; F.Varella & P.Bourgine, pp.134-142, 1991.

COLORNI, A .; DORIGO, M.; MANIEZZO, V.; An Investigation of some properties of an ant algorithm; Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), Brussels, Belgium, R.Männer, B Manderick. p.509-520, 1992.

DE OCA, M. A. M., PEÑA, J., STUTZLE, T., PINCIROLI, C., & DORIGO, M. Heterogeneous Particle Swarm Optimizers. In evolutionary computation, 2009. Cec'09. IEEE Congress on (pp. 698-705). IEEE.

DORIGO,M.; GAMBARDELLA, L. M.; Ant colonies for the traveling salesman problem. Belgium; Université Libre de Bruxelles, 1996

DORIGO, M.; MANIEZZO, V.; COLORNI, A.; Ant System: Optimization by a colony of cooperating agents. IEEE Trans. On Systems, Man and Cybernetics-Part B: Cybernetics, 26, v.1, p.29-41, 1996.

DANTZIG, G. B.; FULKERSON, D. R.; JONSON, S. M. (1954). Solutions of a large scale traveling salesman problem. Operations Research 12, 393-410.

DENNIS JR, J. E.; SCHNABEL R. B. Numerical Methods for Unconstrained Optimization and Nonlinear Equations, capítulo 6, pp. 111-152. Prentice-Hall, 1983.

FEO, T.A.; RESENDE, M.G.C; A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, 8: pp 67-71, 1989.

FEO, T.A.; RESENDE, M.G.C; Greedy randomized adaptive search procedures. Journal of Global Optimization 6, pp 109-133, 1995

EBERHART, R. e KENNEDY, J. A New Optimizer Using Particle Swarm Theory. Em: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, p. 39-43, 1995.

EBERHART, R. C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, v. 1, p. 84–88 vol.1, 2000.

FRAGA, M.C.P.;Uma metodologia híbrida Colônia de Formigas - Busca Tabu Reconexão por Caminhos para resolução do Problema de Roteamento de Veículos com Janelas de Tempo; Dissertação de Mestrado, CEFET-MG, 2006.

GANHOTO, M.A. Abordagens para problemas de roteamento; Dissertação de Mestrado, Universidade Estadual de Campinas, 2004.

GLOVER, F., KOCHENBERGER, G.A. Handbook of Metaheuristics. Kluwer Academic Publishers. Boston, 2003.

GLOVER, F. LAGUNA, M. Tabu Search. Boston, MA: Kluwer Academic Publishers. xix, 1997. 382p.

GLOVER, F. LAGUNA, M. Fundamentals of scatter search and path-relinking. Control and Cybernetics, v.29, n. 3, p 653-684, 2000.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização Combinatória e Programação Linear – Modelos e Algoritmos. Editora Campus. Rio de Janeiro, 2005.

GREFENSTETTE, J.J.; Optimization of control parameters for genetic algorithms; IEEE Transactions on Systems, Man and Cybernetics, v.16, n.1, p.122-128, 1986.

GREIG D. M. Optimization, capítulos 3-4. Longman Inc., New York, USA, 1980.

HEPPNER, F.; GRENANDER, U. (1990). A Stochastic nonlinear model for coordinated bird flocks. In S. Krasner, Ed., The Ubiquity of Caos. AAAS Publications, Washington, DC.

JOHNSON, D. S.; MCGEOCH, L. A. Experimental analysis of heuristics for the stsp. In: Local Search in Combinatorial Optimization. Wiley & Sons, 2001.

JOHNSON, S. Emergence: The Connected Lives of Ants, Brains, Cities, and Software. Scribner, 2001. 288 p.

KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ, USA, 1995. p. 1942–1948.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'. 1997 IEEE International Conference on, v. 5, p. 4104–4108 vol.5, 1997.

KENNEDY, J.; EBERHART, R. C. Swarm Intelligence. Academic Press, 2001. 512 p.

KENNEDY, J.; EBERHART, R. C. (1995). Particle swarm optimization. Neural Networks. Proceedings, IEEE International Conference on. Perth, WA.

KENNEDY, J.; EBERHART, R. C. Swarm Intelligence. Morgan Kaufmann Publishers, California, USA, 2001.

KENNEDY, J.; MENDES, R. Population Structure and Particle Swarm Performance. Em: Proceedings of the IEEE Conference on Evolutionary Computation, p.1671-1676, 2002.

KENNEDY, J.; MENDES, R. Population structure and particle swarm performance. In Congress on Evolutionary Computation, Volume 2, páginas 1671-1676, Piscataway, NJ, Maio de 2002. IEEE Service Center.

KIRKPATRICK, S.; GELATT Jr, C.D.; VECCHI, M.P.; Optimization by Simulated Annealing; Science, Vol 220, N.4598, 1983.

KNOWLES, J.; CORNE, D. Approximating the non-dominated front using the pareto archived evolution strategy. Evolutionary Computation, 8(2):149-172, 2000.

LAUMANNS,M.; ZITZLER, E.; THIELE L. A unified model for multi-objective volutionary algorithms with elitism. Congress on Evolutionary Computation (CEC), páginas 46-53, Piscataway, NJ, 2000. IEEE Service Center.

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. Operations Research, v. 21, n. 2, p. 498–516, 1973.

MALAQUIAS, N.G.L.; Uso dos Algoritmos Genéticos para a Otimização de Rotas de Distribuição; Dissertação de Mestrado. Universidade Federal de Uberlândia, 2006.

MAYERLE, S.F.; Um Algoritmo Genético para o Problema do Viajante. Artigo de Circulação Interna do Departamento de Engenharia de Produção e Sistemas da UFSC, 1994.

MILLONAS, M. M. Swarms, phase transitions, and collective intelligence. No: C. G. Langton, Ed., Artificial Life III. Addison Wesley, Reading, MA.1994.

MITRA, D.; ROMEO, F.; SANGIOVANNI-VINCENTELLI, A.; Convergence and finite-time behavior of simulated annealing; Advances in Applied Probability, N.18, pp.747-771, 1986.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. Combinatorial optimization: algorithms and complexity. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN 0-13-152462-3.

POLTOSI, M. Programando Soluções para Problemas de Otimização Combinatória. No Congresso Simulado de Técnicas de Programação, São Leopoldo, 2006.

POTVIN, J. Y. Genetic algorithms for the traveling salesman problem. Annals of Operations Research 6, p.339 - 370, 1996.

REEVES, C. R. Modern Heuristic Techniques for Combinatorial Problems. New York McGraw-Hill International. 1995. 320p.

REYES-SIERRA, M.; COELLO, C. A. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research, 2006.

RONALD, S. More distance functions for order-based encodings. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. 1998. p. 558–563.

ROSENDO, M.; POZO, A. A hybrid particle swarm optimization algorithm for combinatorial optimization problems. In: IEEE Congress on Evolutionary Computation (CEC 2010). Barcelona, Spain, 2010.

ZITZLER, E.; DEB, K.; THIELE, L. Comparision of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, 8(2):173-195, 2000.

WANG, KANG-PING; HUANG, LAN; ZHOU, CHUN-GUANG; PANG, WEI (2003). Particle swarm optimization for traveling salesman problem. Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an. 1583-1585.
WEST, D. B. Introductoin to Graph Theory (2nd Edition). Prentice Hall, 2000.

WHITE, T.; PAGUREK, B. Towards multi-swarm problem solving in networks. In: In Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98. IEEE Computer Society, 1998. p. 333–340.

WHITLEY, D.; STARKWEATHER, T.; FUQUAY, D.; Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator; Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, 1989.

ZAMBONI, L.V.S.; Técnicas de Roteirização de Veículos Aplicadas ao Transporte Escolar; Dissertação de Mestrado, UFPR, 1997.

ZUBEN, F. J. V.; ATTUX, R. R. F. Inteligência de Enxame. DCA/FEEC/Unicamp e DECOM/FEEC/Unicamp, 2008.